ATARI

ST COMPUTER

Die Fachzeitschrift für ATARI ST- und TT-Anwender

Juli/August 91

DM 8,- Ös. 64,- Sfr. 8,-

7/8

Familienbande

Objekt-orientierte Programmierung in C

Business

K-Spread 4 Tabellenkalkulation

Verwandlung

Spectre 3.0 macht den Atari zum Macintosh

Mammut-RAM

Virtuelle Speicherverwaltung mit dem Atari TT

Digitale Justitia

Sound-Sampling erlaubt?



Der Turmbau zu Babel

Nicht, daß unsere Leser jetzt denken, wir würden ins biblische Zeitalter zurückkehren. Auch wenn vor ca. 2000 Jahren manches anders und vieles besser war (z.B. der Sommer!), möchte ich doch nicht mit unseren Vorfahren tauschen. In der heutigen Zeit werden andere Türme erbaut. Vielleicht nicht bis in den Himmel, aber immer noch hoch genug. Einige davon befinden sich bestimmt in den Computern unserer treuen Leser.
Ich spreche natürlich von den diversen Hardware-Erweiterungen, die den guten alten ST immer mehr in Richtung TT weisen. RAM-Erweiterungen auf 4MB sind mittlerweile schon fast Standard. Turbo-Karten mit 16, 32 oder 50 MHz Taktrate, 68020- oder gar 68030-Prozessor (apropos... wo bleibt der 68040?) machen der „gebeutelten" ST-Hardware immer mehr „Feuer unter dem H.....". Natürlich sollte auch eine Grafikerweiterung her, 16 Millionen Farben in immenser Auflösung sind heute nicht mehr dem Profi aus dem Fernsehstudio oder der CAD-Maschine vorbehalten. Fehlt nur noch ein ordentlicher MS-DOS-Emulator (man sollte schon kompatibel zur guten alten Zeit sein), und schon erreicht der Turm im ST schwindelnde Höhen, und das überlastete Netzteil singt vor Freude ganze Arien.
Fragt man sich nun nach Sinn oder Unsinn solcher Erweiterungen (schließlich bietet der TT fast alles serienmäßig in einem Gehäuse), stellt sich im Gespräch mit Benutzern solcher Maschinen heraus, daß einige von einer gewissen Leidenschaft besessen sind, aus dem betagten „Maschinchen-ST" doch noch eine Super-Workstation zu machen und damit die Zeit und den technologischen Fortschritt ein wenig einzuholen. Der Großteil versucht aber, dadurch Schritt für Schritt dem Flair eines TT näherzukommen, ohne gleich die „Radikalkur" anzuwenden und den ST zum alten Eisen zu werfen. Immerhin wurden (und werden noch!) Computer der ST-Serie allein in Deutschland viele tausend Mal verkauft. Wenn also schon ein ST im Hause ist, dann kann er auch mit allen erdenklichen „Tuning-Maßnahmen" bestückt werden. Das muß ja nicht von heute auf morgen passieren. Der TT läßt sich halt leider nicht in kleinen „Häppchen" Stück für Stück kaufen.

Christian Möller

INHALT

SOFTWARE

HARDWARE

ST-REPORT

GRUNDLAGEN



Objektorientierte Programmierung in C

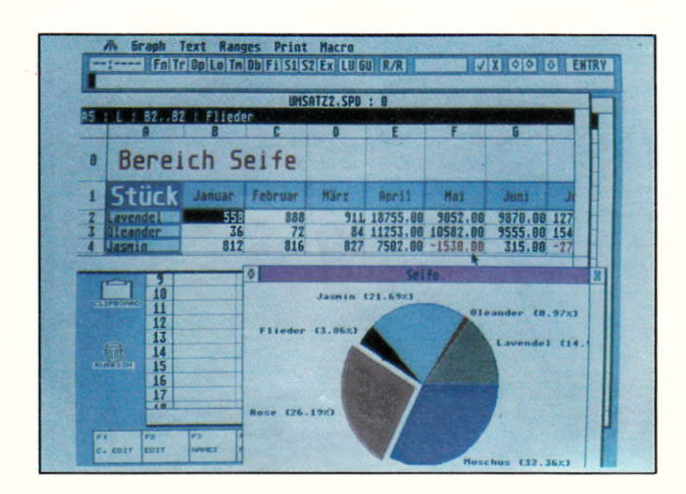
Jeder, der schon mal das Vergnügen hatte, größere Programme zu schreiben, wird sicher zugeben, daß sich gewisse Strukturen, seien es Algorithmen oder Datenverbände ständig nahezu unverändert in seinen Werken wiederholen. Leider nur nahezu. Die kleinen Unterschiede geben dem Speicherplatz- und Rechenzeitbewußten genug Anlaß, jedesmal eine neue Implementation vorzunehmen. Die objektorientierte Programmierung (OOP) führt zusätzlich zu den uns schon bekannten Sprachkonstrukten das Objekt ein, in dem eine Sammlung von Daten und Strukturen vereinigt sind.

Seite **104**

Spectre 3.0 - Der Atari wird zum Mac

Den letzten Bericht über den Spectre, den Emulator, der den Atari-Rechnern das Leben eines Apple Macintosh einhaucht, konnten Sie bei uns vor einiger Zeit lesen (genau genommen in der Mai-Ausgabe '89). Seitdem hat sich einiges getan. Zunächst einmal ist die deutsche Konkurrenz des Spectre, Aladin, aufgrund von Rechtsstreitigkeiten mit Apple vom Markt verschwunden. Es hat sich mal wieder gezeigt, daß Apple nicht sehr gut auf Clones und Emulatoren zu sprechen ist. Doch der Spectre scheint damit bis jetzt noch keine Probleme zu haben. Er liegt mittlerweile in der Version 3.0 vor und läuft auf ST, STE und TT.

Seite **44**



K-Spread 4 - Tabellenkalkulation

Tabellenkalkulationen sind heute aus dem logistischem Bereich eines Büros nicht mehr wegzudenken. Da werden Jahrespläne in Formeln geschmiedet und anschauliche Grafiken über Verkaufszahlen erstellt. Wir wollen Ihnen die deutsche Version von K-Spread 4, einer Tabellenkalkulation der englischen Firma Kuma vorstellen, die jetzt über Omikron vertrieben wird.

Seite 20



Die digitale Justitia - Sound-Sampling

Die Geschichte der Kunst ist auch eine Geschichte der Plagiate. Angefangen von Variationen bekannter Klassiker (z.B. Brahms über ein Thema von Joseph Haydn {op.56}), zieht sich der rote Faden des Ideenklaus durch die Musikbranche bis zur heutigen Zeit. Die Technik macht jedoch alles einfacher. Mußte man sich bei einem Plagiat früher noch Gedanken darüber machen, wie der Urheber die Noten gesetzt hat, so erledigt dies heutzutage der Computer selbst. Das Zauberwort hierzu heißt: SAMPLING. Wir haben für Sie den rechtlichen Hintergrund beleuchtet.

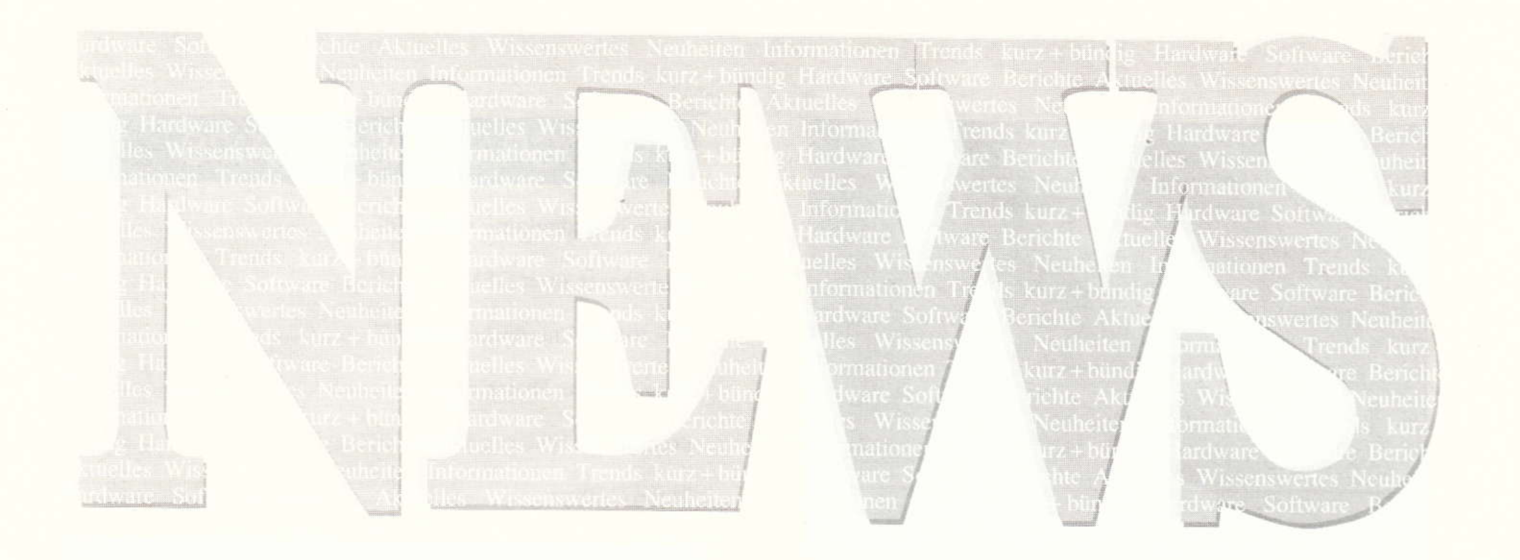
Seite 62

PROGRAMMIERPRAXIS

AKTUELLES

PUBLIC DOMAIN

RUBRIKEN



NEWS

Düsseldorf

ATARI messe

23. bis 25. August 1991

Atari-Messe 1991

Auch dieses Jahr öffnet die „größte Herstellermesse Europas", die Atari-Messe, wieder ihre Pforten. Wie immer findet sie in Düsseldorf statt. Der Termin ist auf den 23. bis 25. August 1991 festgelegt. Insgesamt 220 Aussteller bieten auf 20.000 Quadratmetern Fläche Neuheiten in Soft- und Hardware rund um die Atari ST- und TT-Computer an. Atari erwartet ca. 50.000 Besucher, die sich auf ein buntes Spektrum an Workshops, Sonderveranstaltungen, Vorführungen und Neuheiten freuen können. Natürlich werden auch der Heim Verlag und die MAXON Computer GmbH mit ihren Ständen zu finden sein. Dies wird die nunmehr 5.Atari Messe werden, was eindeutig für die Konstanz und den Erfolg der Atari-Produkte spricht. Man darf gespannt sein!

Augur Tool für Syntex

Der Bibliothekseditor „Augur-Tool", der zum Lieferumfang von Augur gehört, ist jetzt auch als Zusatz zum weitverbreiteten OCR-Programm Syntex der Marvin AG erhältlich. Dieses Werkzeug erlaubt die nachträgliche Pflege von Schriftbibliotheken. Neben der Korrektur von falsch zugeordneten Zeichen ist das Löschen und Einfügen von Zeichen in eine Bibliothek möglich. Selbst das Mischen zweier unterschiedlicher Bibliotheken ist machbar. Vor dem Speichern wird die Bibliothek optimiert, um zukünftige Erkennungsvorgänge weiter zu beschleunigen. AugurTool für Syntex, Preis sFr. 90,-, eine lohnende Investition für Syntex-Besitzer.

Trillian Computer AG
Eisfeldstraße 6
CH-8050 Zürich
Tel.: 01/3022179

Kostenlose Kleinanzeigen ade!

Leider ist es uns in Zukunft nicht mehr möglich, unseren Lesern den Service der kostenlosen privaten Kleinanzeigen anzubieten. Begründet ist dies in einem Urteil des Bundesverfassungsgerichts in Karlsruhe. Dort steht geschrieben, daß kostenlose Veröffentlichung privater Kleinanzeigen in Fachzeitschriften in der Regel wettbewerbswidrig sind. Ein derartiges Angebot bedeute eine „Gefährdung" des Wettbewerbs auf dem prinzipiell „geschlossenen Markt der Fachzeitschriften" (AZ: I ZR 55/89 vom 14. März 1991). Aus diesem Grund können in Zukunft in der ST-Computer private Kleinanzeigen nur noch gegen einen Unkostenbeitrag von DM 2,- in Briefmarken veröffentlicht werden. Andernfalls würden wir das Risiko einer Abmahnung und ein Nichterscheinen der ST-Computer riskieren.

Ihre Redaktion

IRNET - ein optisches Netzwerk für Druckerbetrieb

In vielen modernen Büros verrichten heutzutage mehrere PCs ihre Dienste. Um von jedem dieser PCs aus drucken zu können, ist ein erheblicher Verkabelungsaufwand nötig. Die von der Firma Schirmer GmbH, Bielefeld, entwickelten IRNET-Modems werden an PCs, Drucker oder Plotter angeschlossen und bilden ein drahtloses Datenübertragungsnetzwerk mit bis zu 16 Geräten, in dem Daten von jedem PC zu jedem Drucker wahlfrei übertragen werden können. Die überbrückbare Entfernung zwischen PC und Drucker beträgt ca. 10 Meter. Ein spezielles Übertragungsprotokoll verhindert Datenverluste bzw. Datenkollisionen. Jedes IRNET-Modem ist in der Lage, ca. 32 kB (64 kB) Druckzeichen zwischenzuspeichern, um Übertragungsengpässe zu verhindern und die PCs einsatzbereit zu halten, auch wenn ein Druckauftrag eines anderen PCs bearbeitet wird. Vor jedem Druckauftrag kann der Benutzer einen der am Netz teilnehmenden Drucker als Zielgerät auswählen. Die IRNET-Modems sind ausgelegt für den Anschluß an die V24- oder Centronics-Schnittstelle eines PC oder eines Atari ST/TT. Die Übertragungsgeschwindigkeit beträgt 9600 Baud. Im Lieferumfang enthalten ist ein Accessory für Atari und wahlweise ein V24- oder ein Centronics-Kabel für den PC/ST. Eine ZZF-Zulassung ist beantragt. Der Verkaufspreis liegt je nach Version und Speicherausbau zwischen DM 430,- und DM 500,-.

Schirmer GmbH technisches Büro
Strothbachstraße 11
4800 Bielefeld 11

Taskhelp V 3.04

Das Multi-Accessory Taskhelp liegt nun in der Version 3.04 vor. Neben den üblichen Funktionen wie Maus-Speeder, Uhr mit Wekker, Dateifunktionen wie Löschen, Kopieren, Umbenennen usw. bietet es auch einen Event-Recorder zum Aufzeichnen und Wiedergeben von Maus- und Tastaturaktionen sowie eine einstellbare Druckeranpassung. Hervorzuheben ist auch eine Funktion zum Formatieren von Disketten im Hintergrund, d.h. daß der Anwender normal mit seinem Programm weiterarbeiten kann, während Taskhelp gerade eine Diskette formatiert. Die neue Version arbeitet nun auch mit beweglichen Dialogen und ist per Tastenkombination aufrufbar.

Stuhr & Jacobs
Dieselstraße 9
W-6100 Darmstadt
Tel.: 06151/98406-0

Skyplot Plus für den TT

Die Anpassung vorhandener ST-Software an den TT geht weiter. Mit Skyplot Plus 3d wird jetzt auch dieses bekannte Programm für „Himmelsgucker" auf dem TT lauffähig. Dabei unterstützt die neue Version auch den mathematischen Coprozessor des TT. Da die Berechnung von Sternenpositionsdaten hauptsächlich auf trigonometrischen Funktionen beruht, kann der Coprozessor hier einen erheblichen Geschwindigkeitszuwachs erzielen (ca. Faktor 10!). Zudem sind die Berechnungen auch noch wesentlich genauer! Die TT-Version wird für 298,- DM verkauft. Eine spezielle Version zum direkten Steuern von CCT-Teleskopen ist nun auch verfügbar. Damit sind den Sternenfreunden wohl nur noch die Wolken im Weg.

Heim Verlag
Heidelberger Landstraße 194
W-6100 Darmstadt
Tel.: 06151/56057

Adimens für Druckereien

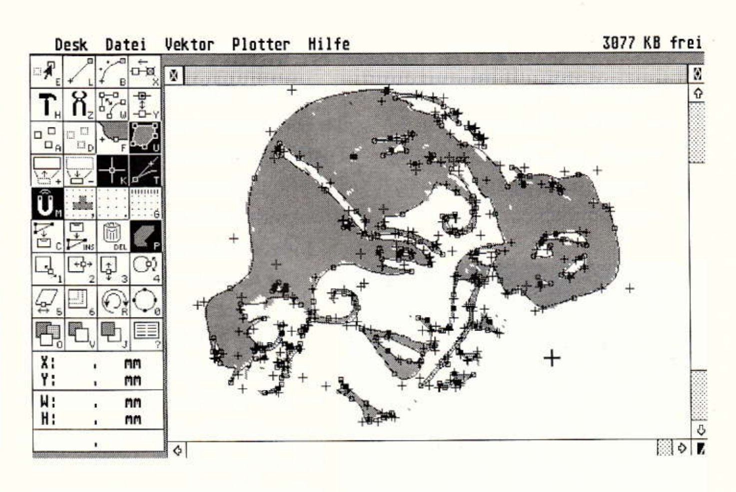
Aufbauend auf dem Datenbankprogramm Adimens ST, gibt es ein Kalkulationsprogramm für Druckereien unter den Namen „DRUCKEREImens". In der jetzt aktualisierten Version Nr. 3.0 bietet es eine umfassende Angebots- und Auftragskalkulation für alle Druckwerke getrennt für Umschlag und Inhalt in 1- bis 4-Farbdruck sowie Zusatzfarben der Europalette. Eine zusätzliche Kalkulation für Papier-, Druck- und Druckweiterverarbeitung entspricht den Richtlinien des Bundesverbandes Druck e.V. Spezialanpassungen auf Kundenwunsch sind jederzeit möglich. DRUCKEREImens kostet 998 DM.

comtex Computersysteme
Gitteweg 3
W-7801 Bollschweil
Tel. 07633/50784

Umdenken bei TommySoft

TommySoft ändert seine Produktpalette. Ab sofort sind die Programme SoundMerlin, SoundMachine und GraphStar aus dem Vertrieb genommen. Sie werden jetzt als Shareware-Versionen freigegeben. Handbücher für diese Produkte sind nur noch per Postweg erhältlich. TommySoft begründet diesen Schritt mit der mangelnden Marktresonanz, besonders für die Sound- und Musikprogramme. Für die Atari-Messe kündigt TommySoft die Version 4.0 ihres Zeichenprogrammes „MegaPaint" an, das mit zahlreichen Neuerungen und Erweiterungen aufwartet. Es ist nun generell auf Rechnern der TT-Serie einsetzbar und bietet u.a. auch überarbeitete und neue Druckertreiber.

TommySoftware
Selchower Strasse 32
W-1000 Berlin 44
Tel.: 030/621-4063



Nachtrag zu AVANT-Vektor

Einige Berichtigungenbzw. Neuigkeiten zum Test des Vektorisierungs-Utilitys „AVANT-Vektor" (s.Ausgabe 6/91) liegen vor: Das Programm ist auch in der Lage, monochromes TIFF-Format zu lesen. Außenlinien lassen sich vektorisieren, und eine Linienstärke ist ebenfalls einstellbar. Im Test wurde beanstandet, daß sich die Parameter der Vektorisierung nicht fein genug einstellen lassen, um bis auf Pixel-Größe hinunter zu konvertieren. Wie uns die Firma „Trade It" mitteilte, ist es aber doch möglich. Ab der neuen Version 1.2 sind auch neue Features hinzugekommen, wie unterschiedliche Linienstärken pro Objekt und numerische Eingabe von Parametern. Probleme, die es mit dem Atari TT und dem Laserdrucker gab, sollen mittlerweile beseitigt worden sein. Ebenso hat man nun eine Ansteuerung von GP-GL-Plottern (Schneid-Plotter) integriert. Die neue Version 1.2 ist in drei Ausbaustufen erhältlich. Die Grundversion; „AVANT-Trace" ist für 298,- DM erhältlich- die bekannte Version mit den o.a. Neuerungen kostet nun 698,- DM. Hinzugekommen ist auch eine Profi-Version mit dem Namen: „AVANT-Plot". Diese beinhaltet eine Schnittstelle zu EPS (Encapsulated Postscript) und besitzt spezielle Funktionen für den Umgang mit Schneid-Plottern. Alle neuen Versionen sind ab sofort für registrierte Benutzer als Update verfügbar.

Trade It
Jahnstraße 18
W-6112 Groß-Zimmern
Tel.: 06071/41089

Artworks im Süden

Die Firma Duffner Computer, Freiburg, hat ab sofort den Vertrieb von „Artworks Business" und „Artworks Fonts" für den Süddeutschen Raum übernommen. Des weiteren kommt aus der Artworks Collection ein weiteres Bonbon: Plotter-Fonts für den Atari ST! In ein und denselben Font kann man mit beliebigen Farben, Rastern, Outlines und Farbflächen bearbeiten. Die Fonts „Headline", „Headline rund" und „fine shadow" sind ideal geeignet für paßgenauen Mehrfarbendruck und für den Folienplotter.

Duffner Computer
Habsburger Straße 43
W-7800 Freiburg
Tel. 0761 56433

Nachtrag zum BASIC-nach-C-Konverter

Leider wurde beim Test des „CICERO-BASIC-nach-C-Konverters" in der vergangenen Ausgabe versäumt, Bezugsadressen für das Programm anzugeben. Dies holen wir hiermit nach. Der Konverter ist zu beziehen bei:

CICERO Otto Vinzent
Ballweilerstraße 7
W-6676 Mandelbachtal 4
Tel.: 06803/2834

oder

H.Richter Distributor
Hagener Straße 65
W-5820 Gevelsberg
Tel.: 02332/2706

oder:

Maranatha
Albertstraße 26
W-6750 Kaiserslautern
Tel.:0631/26656

TT in neuem Gewand

Ein brandneues Design bietet die Firma TETRA-Computer für ihr Flaggschiff, den Tetra 030 an. Der sehr futuristisch wirkende TT-Tower wartet mit einem High-Tech-



Innenleben auf: 8 MB RAM, 213 MB Festplatte, 68030-Prozessor und 32 MHz Systemtakt sind nur einige der zahlreichen Features dieses auf TT-Basis erweiterten Computers. Als Tastatur setzt Tetra ein Qualitätsprodukt aus dem PC-Bereich ein, das 100% kompatibel zur original TT-Tastatur ist. Dies wurde durch eine eigene Tastaturschnittstelle realisiert. TTL-Schaltausgänge sowie ein 8-Bit-Analog/Digital-Wandler-Eingang machen diesen Computer ideal für industrielle Anwendungen in der Steuerungstechnik. Solch geballte Computerpower hat natürlich ihren Preis. Ab 9980,- DM ist dieser Spezial-TT zu haben. Wer will, kann sich gleich diverse Zusatzoptionen (gegen Aufpreis) einbauen lassen, als da wären: SyQuest-44MB-Wechselplatte, Teac-155MB-Streamer, Großmonitor etc. Ausgeliefert wird das 21 kg schwere Schmuckstück mit eigener Software zur Ansteuerung der neuen Schnittstellen. Dies schließt auch eine XBIOS-Erweiterung des Betriebssystems ein, so daß Entwickler sich sofort auf die Verwendung der zusätzlichen I/O-Ports einstellen können.

Tetra Computersysteme GmbH
Neuer Markt 27
W-5309 Meckenheim
Tel.: 02225/17081



Neuer BTX/VTX-Manager von Drews

Drews liefert ab sofort die Version 4.0 des bekannten BTX/VTX-Managers aus. Die wichtigsten Neuerungen sind: Anpassung an TT sowie an Großbildschirme, konsequente Einbindung in GEM und ein komfortabler Makroeditor, der auch das Erstellen und Ändern von Makros als ASCII-Text erlaubt. Der neuen Version liegt nun auch ein Buch mit dem Namen: „ST online" bei. Dieses Buch ist neben Dokumentation zur eigentlichen Software auch hilfreicher Ratgeber und Führer durch den Dschungel des BTX-Systems. Gerade dem BTX-Neuling wird durch das Buch der Einstieg in die Welt der Datenfernkommunikation erheblich erleichtert. Auf 200 Seiten bekommt der Anwender Tips zum Anschluß seines BTX-Modems und wichtige Hintergrundinformationen rund um das Thema BTX geboten. Die neue Version 4.0 ist für 149,-DM (an Hayes-Modem) bzw. 229,- DM (an DBT03) zu kaufen.

Drews EDV + Btx GmbH
Bergheimer Straße 134b
Postfach 101806
W-6900 Heidelberg
Tel.: 06221/29900

Mit dem ST Grammatik lernen

Bislang war der Computer als Instrument zum Sprachenlernen lediglich als Vokabeltrainer einsetzbar. Das eigentlich Schwierige an fremden Sprachen, die Grammatik, mußten die Lernwilligen immer noch „von Hand" in den Kopf befördern. Doch auch hier schreitet die Entwicklung voran. Ein Programm der Firma „Falken-Software" verspricht Abhilfe. „The Grammar Master" ist der bezeichnende Titel dieses Software-Produktes, und in der Tat, selbst mit geringen Englischvorkenntnissen kann man sehr schnell den Eigen- und Besonderheiten der englischen Grammatik auf die Spur kommen. Zwar ist die Bedienung ausschließlich per Tastatur zugelassen (PC-Portierung?), aber da das Programm des öfteren sogar ganze Satzkonstruktionen vom Benutzer verlangt, ist dieses Manko nicht weiter schwerwiegend. Der geneigte Benutzer wird einfach und dialogorientiert mit den verschiedenen grammatikalischen Regeln vertraut gemacht. Dabei geht das Programm nach dem Verfahren: „Versuch und Irrtum" und „Multiple Choice" vor. Zu den ausgewählten Fragen bietet es meist mehrere Antwortmöglichkeiten an und schreitet ein, sobald der Benutzer einen Fehler macht. Er hat dann die Möglichkeit, Genaueres über die verletzte Grammatikregel zu erfahren und diese speziell anhand von weiteren Beispielen zu üben. Der Schwerpunkt liegt dabei auf der Behandlung der „Tenses", also der Zeitformen. „The Grammar Master" läuft auf dem Atari ST in der hohen und mittleren Auflösung und auf dem TT (nur ST-Auflösung!)

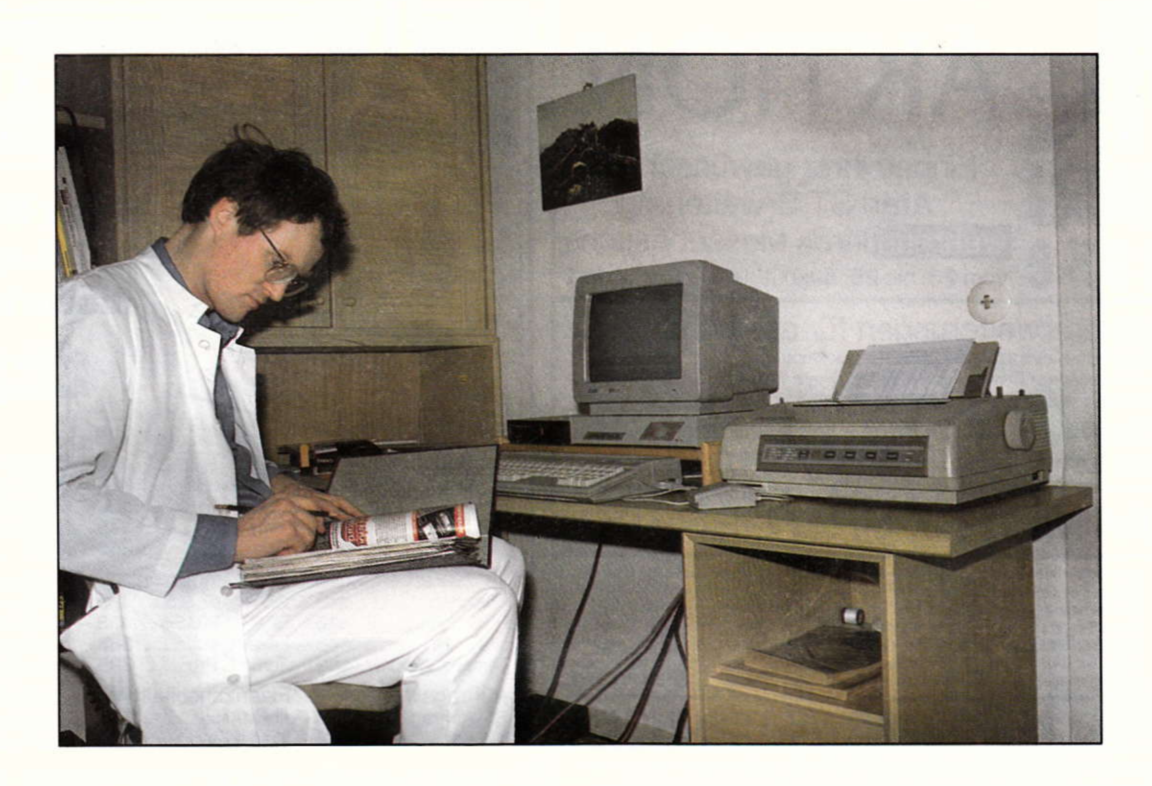
Falken Verlag GmbH Software
Schöne Aussicht 21
W-6272 Niedernhausen
Tel.: 06127/7020

SM-124-Emulator für den TTM-194

Ein sehr nützliches Utility für TT-Besitzer, die ausschließlich den TT-Großmonitor TTM-194 benutzen, bietet die Firma Overscan an. Mit dem 99.- DM kostenden SM-124-Emulator ist es möglich, auch in der hohen TT-Auflösung Programme wie z.B. Signum!, oder STAD zu benutzen. Der Emulator reduziert dabei die hohe Auflösung von 1280x 800 Punkten auf die ST-übliche Auflösung von 640x400 Punkten. Dabei kann dennoch der komplette TT-Bildschirm gefüllt werden (Zoom Modus). Die Umschaltung auf diese Auflösung erledigt der Emulator automatisch anhand einer Liste, in der die problematischen Programme eingetragen werden. Diese Liste kann auch mit Hilfe eines CPX-Moduls für das TT-Kontrollfeld angepaßt, bzw. erweitert werden.



Overscan GbR
Säntisstraße 166
W-1000 Berlin 48
Tel.: 030/7219466



Rasterfahndung gegen Bakterien

Infektionen aufspüren mit dem Atari

Wenn das Bundeskriminalamt Terroristen per Rasterfahndung im Computer suchte, so war das nicht bei allen Menschen unumstritten. Wohl niemand dürfte sich jedoch daran stören, wenn der Atari, auf dem gleichen Prinzip basierend, gefährliche Bakterien als Erreger von Krankenhausinfektionen jagt. Genau dies geschieht mit dem Programmpaket MIKRO.DAT an einer mittelgroßen Klinik in Bayern. MIKRO.DAT wurde speziell für diesen Anwendungsfall programmiert, da ein vergleichbares Programm nicht anderweitig verfügbar ist.

Damit Nichtmediziner jetzt nicht gleich gelangweilt weiterblättern, zunächst einige Ausführungen zu Sinn und Zweck des Programmes. An unserem Krankenhaus fallen pro Jahr wie in allen vergleichbaren Kliniken ca. 10000 mikrobiologische Untersuchungen an. Das heißt, es werden Proben von Patienten entnommen (z.B. von Wunden, Blut, Urin, anderem Gewebe usw.) und dann untersucht, welche Bakterien in diesen Proben enthalten sind. Dies geschieht meist nicht direkt unter dem Mikroskop, sondern durch den Versuch, die Bakterien auf speziellen Nährböden anzuzüchten. Nach einigen Tagen wird dann die Bakterienart mit bestimmten Methoden identifiziert. Und - ganz wichtig - dann wird durch Zugabe von Proben einzelner Antibiotika getestet, welches dieser Medikamente gegen das jeweilige Bakterium wirksam ist.

Die gesamte Prozedur bis zum Vorliegen der Wirksamkeitstestung dauert allerdings mindestens einige Tage, bis dahin könnte der Patient schon an seiner Infektion gestorben sein! Die Behandlung muß also manchmal schon sofort einsetzen. Dann richtet sich der Arzt nach Erfahrungswerten. So weiß er, daß bei bestimmten Krankheitsbildern meist Penizillin hilft, bei anderen eher eines der neueren Antibiotika. Die Häufigkeit bestimmter Bakterienarten und ihre Empfindlichkeit gegen Antibiotika ist jedoch nicht überall und immer gleich. Vielmehr hat jede Klinik ihre speziellen Problembakterien, die immer wieder schwere Infektionen auslösen und besonders schlecht mit Antibiotika abzutöten sind. Außerdem können diese


ATARI®
messe
vom 23. bis 25. August 1991
Düsseldorf Messegelände
Hallen 11 und 12 • Täglich 10.00 bis 18.00 Uhr
• Das DTP Center: vom Konzept bis zum Auflagendruck
• MIDI Sonderveranstaltung: Sound und Musik mit dem ATARI ST
• Software rund um den Portfolio
• Neue Datenbanken
• Computer in der Schule – live
• Workshops und Vorträge zu aktuellen Themen aus Wissenschaft, Technik und Ausbildung
• Soft- und Hardware - Anbieter aus Europa und Übersee
• VME: Die Schnittstelle im ATARI Mega STE und TT
• Neues von und mit dem Lynx


ATARI®
messe
vom 23. bis 25. August 1991
Düsseldorf Messegelände
Hallen 11 und 12 • Täglich 10.00 bis 18.00 Uhr
• Das DTP Center: vom Konzept bis zum Auflagendruck
• MIDI Sonderveranstaltung: Sound und Musik mit dem ATARI ST
• Software rund um den Portfolio
• Neue Datenbanken
• Computer in der Schule – live
• Workshops und Vorträge zu aktuellen Themen aus Wissenschaft, Technik und Ausbildung
• Soft- und Hardware - Anbieter aus Europa und Übersee
• VME: Die Schnittstelle im ATARI Mega STE und TT
• Neues von und mit dem Lynx

wissenschaftliche Publikationen. Und wem das noch immer zu trocken ist, der packt die Tabelle in eine spezielle Datei, die vom zugehörigen Grafikprogramm geladen wird, das dann die Daten anschaulich als Blockdiagramm (zweidimensional) darstellt. Dieses Grafikprogramm kann aber auch für andere Grafiken (Torten, Linien) für ganz andere Zwecke eingesetzt werden. Es wurde ja auch ursprünglich für die grafischen Darstellungen in einer Doktorarbeit programmiert.

Doch wie macht man nun dem BKA Konkurrenz bei der Rasterfahndung? Nun, man sucht sich im Hauptmenü per Maus oder Tastatur Kriterien aus, nach denen gerastert werden soll. So z.B. einzelne oder Gruppen von Stationen, Untersuchungsmaterialien (z.B. nur alle Proben aus der Luftröhre oder aus Blut), Geschlecht der Patienten, Alter der Patienten (von ... bis), Zeitraum (z.B. nur einzelne Monate). Und was besonders wichtig ist, man kann auch nur nach Untersuchungen von Patienten mit ausgewählten Diagnosen suchen lassen, wobei die Diagnosenbegriffe vom Benutzer frei gewählt werden. Alle diese Suchkriterien lassen sich frei kombinieren, wodurch sich z.T. sehr differenzierte Fragen beantworten lassen.

So sucht der Computer in Sekundenschnelle alle Patienten heraus, die auf der Intensivstation im 1. Quartal wegen einer Lungenentzündung beatmet werden mußten, älter als 60 Jahre und männlich waren. Daraus kann man dann ersehen, mit welchen Bakterien bei diesen Patienten am ehesten zu rechnen ist und welche Antibiotika am besten helfen. Dies kann manchmal einen lebensrettenden Zeitgewinn bedeuten, bis das Ergebnis der Untersuchungen für einen einzelnen Patienten nach Tagen vorliegt! Wenn dann diese Rasterung für fortlaufende Quartale durchgeführt wird (was das Programm komfortabel unterstützt), kann man auch zeitliche Trends erkennen. So kann man rechtzeitig etwas unternehmen, wenn ein Bakterium überhandnimmt oder ein Antibiotikum immer schlechter wirkt.

(Fast) alles ist möglich

Natürlich kann man auch ziemlich unsinnige Rasterkombinationen vorgeben. Dann bekommt man ein zwar korrektes, aber wenig sinnvolles Ergebnis. Und wie mit jeder Statiskik läßt sich auch damit viel Unsinn treiben. So stellt auch weiterhin der Arzt die Diagnose bzw. entscheidet sich für ein Antibiotikum, und nicht unser Atari, auch wenn wir ihm sonst so viel vertrauen.

Das Programm läuft nun schon seit ca. 18 Monaten und hat uns manches interessante Ergebnis geliefert. In dieser Zeit wurden wiederholt Änderungen daran vorgenommen. Insbesondere haben wir weitere Funktionen eingebaut, wenn uns diese wichtig und machbar erschienen. So hat es unterdessen einen gewissen Reifegrad erreicht. Seine Funktionen entsprechen den praktischen Anforderungen und nicht nur theoretischen Überlegungen. Seit kurzem ist auch die Anpassung an die Gegebenheiten anderer Krankenhäuser bzw. bei Änderungen im eigenen Haus leicht mit Hilfe eines weiteren kleinen Zusatzprogrammes möglich.

Es geht nicht ganz ohne Probleme

Bei aller Begeisterung bleiben jedoch auch Probleme. Das größte besteht im hohen Aufwand an Tipparbeit bei der Eingabe in Anbetracht von ca. 10000 Befunden pro Jahr. Deshalb wurde darauf geachtet, daß wirklich nur ein Minimum an Tastendrücken pro Befund nötig ist. So dauert die Eingabe eines Befundes für den etwas eingearbeiteten Nutzer nur wenige Sekunden. Auch die beim ST-Computer so wichtige Maus kommt im Auswertungsprogramm voll zu Ehren, sie hilft bei der Auswahl aus den Menüs, alternativ kann aber meist auch eine Taste gedrückt werden. Und so richtig austoben kann sich jeder Nagerliebhaber im zugehörigen Grafikprogramm, denn das wurde voll unter GEM geschrieben. Schön wäre jedoch die Dateneingabe über strichcodemarkierte Laborzettel. Doch dann ergäbe sich das neue Problem, ein Strich-Code-Lesegerät per Programm abzufragen. Dazu fehlen uns die Erfahrungen. Außerdem müßten die Diagnosen weiterhin per Hand eingetippt werden, da es sich ja um frei vom Benutzer gewählte Begriffe und nicht nur um eine Auswahl aus einer kleinen Liste handelt.

So wird denn trotz hervorragender Unterstützung durch den ST-Computer weiterhin eine Menge Fleiß nötig sein, bevor der Preis der Erkenntnis aus der Rasterfahndung winkt. Und noch etwas bleibt: Gegen Computerviren, die sich an unserem Atari zu schaffen machen, hilft das Programm leider auch nicht!

Dr.med. Manfred Kester
Am Römerkastell 1
W-6350 Bad Nauheim





Kommunikation des Unbewußten

Belanglos nennen wir ihn Kommunikation, den Informationsaustausch, wenn zwei Menschen miteinander reden. Für uns ist es DIE Kommunikation schlechthin. Als Computernutzer kennen wir aber noch eine andere Form des Informationstransportes; er läuft über die sogenannte Mensch-Maschine-Schnittstelle. Das ist hauptsächlich der Fall, wenn uns irgendwelche Software Hinweise zu ihrer Arbeit auf den Bildschirm bringt.

Wohin die Informationen beim Menschen schließlich gelangen, ist uns ebenso geläufig: das Bewußtsein nimmt sie auf. Jener Ort in unserem Gehirn, der unmittelbar unser Handeln und Denken steuert, nimmt diese bewußten Informationen auf und läßt uns ebenso bewußt darauf reagieren. Lange schon streitet sich die Geisteswissenschaft, in den Disziplinen Medizin, Psychologie, Soziologie bis hin zu Randgebieten der Schulweisheit, ob es auch eine Wahrnehmung außerhalb der Kontrolle unseres Bewußtseins geben kann. Sehr schnell ist der Beweis, ob es eine unbewußte Wahrnehmung gibt, in den Bereich der Scharlatanerie und der Jahrmarktssensationen entglitten.

Geheime Verführer

Tatsache ist, daß es geglückte Experimente gibt, die auf eine unterschwellige, nicht wahrnehmbare Kommunikation hinweisen. Schon im Laufe der 50er Jahre wies Prof. Vance Packard in einem Buch auf die 'nicht bewußte Information' in der Werbung hin, Titel des Buches in Deutschland: „Die geheimen Verführer". In eben diesem Buch berichtete er von einem Experiment, wo bei einer Kinoveranstaltung das kurzzeitige Einblenden von Eiscremewerbung zu einem ungeahnten Anstieg des Speiseeisumsatzes führte. Einen ähnlichen Effekt machen sich die amerikanischen Fernsehsender heute zunutze, um Filmbeiträge an besonders sensiblen Handlungspunkten zu unterbrechen und einen Werbespot einzuspielen. Kurz vor einem Spannungshöhepunkt soll die Aufnahmefähigkeit am größten sein, ein Produkt bleibt länger in Erinnerung. Die Amerikaner treiben es sogar schon zum Exzeß, indem sie einen Film fast im Minutenabstand für Sekunden unterbrechen, um ihre Werbebotschaften zu plazieren.

Zur selben Zeit, als Prof. Packard mit seinen Experimenten an die Öffentlichkeit trat, sind auch Versuche in ähnlicher Richtung von dem amerikanischen Marktforscher James Vicary bekannt geworden. Er soll nach eigenem Bekunden während eines Kinofilms im Abstand von mehreren Sekunden Schrifttafeln mit der Aufforderung „Trinkt Cola" oder „Eßt mehr Popcorn" jeweils in einer Dauer von nur einer Dreitausendstelsekunde auf die Leinwand projiziert haben. Der Popcorn-Umsatz stieg um fast die Hälfte an, und Cola-Limonade wurde um fast 20% mehr verkonsumiert.

Ein Fall politischer Einflußnahme wurde vom letzten Wahlkampf zur französischen Präsidentschaft bekannt. Es sollen dreimal am Tage, über drei Monate hinweg, im Indikativ (Vorspann) einer Nachrichtensendung Bilder von François Mitterand so extrem kurzzeitig in das Sendebild eingeblendet worden sein, daß eine bewußte Realisierung durch die Zuschauer nicht möglich war. Glücklicherweise ist eine solche Stimulation der Fernsehzuschauer in Deutschland für jede Art von Werbung grundsätzlich verboten worden.

Es gibt bereits eine regelrechte Industrie, die sich dieser unterschwellingen Informationsvermittlung bedient. In jedem Kaufhaus plätschern (fast) unhörbar Melodien auf uns nieder, die einerseits die Kauflust steigern, andererseits den Kaufhausdiebstählen entgegenwirken sollen. Auch in der Musik gibt es unzählige Beispiele unbewußter Informationsübermittlung. Neben hörbaren Melodien liegt eine Textinformation so leise darunter, daß sie die Wahrnehmungsschwelle nicht erreicht. Selbst kein Geringerer als Alfred Hitchcock hatte sich in der Erstverfilmung seines Streifens 'Psycho' dazu hinreißen lassen, die Worte „Messer", „Mord" und „Blut" an geeigneter Stelle unter die (laute) Musik flüstern zu lassen, um die Spannung unter den Zuschauern zu steigern.

Für alle Beispiele, ob Texttafel, Bild oder Musik, gibt es eine Gemeinsamkeit. Die unbewußt übermittelte Information wird zwar an den menschlichen Empfänger übertragen, aber aufgrund besonderer Umstände (zu kurzzeitig, zu leise) dem Bewußtsein

wissenschaftliche Publikationen. Und wem das noch immer zu trocken ist, der packt die Tabelle in eine spezielle Datei, die vom zugehörigen Grafikprogramm geladen wird, das dann die Daten anschaulich als Blockdiagramm (zweidimensional) darstellt. Dieses Grafikprogramm kann aber auch für andere Grafiken (Torten, Linien) für ganz andere Zwecke eingesetzt werden. Es wurde ja auch ursprünglich für die grafischen Darstellungen in einer Doktorarbeit programmiert.

Doch wie macht man nun dem BKA Konkurrenz bei der Rasterfahndung? Nun, man sucht sich im Hauptmenü per Maus oder Tastatur Kriterien aus, nach denen gerastert werden soll. So z.B. einzelne oder Gruppen von Stationen, Untersuchungsmaterialien (z.B. nur alle Proben aus der Luftröhre oder aus Blut), Geschlecht der Patienten, Alter der Patienten (von ... bis), Zeitraum (z.B. nur einzelne Monate). Und was besonders wichtig ist, man kann auch nur nach Untersuchungen von Patienten mit ausgewählten Diagnosen suchen lassen, wobei die Diagnosenbegriffe vom Benutzer frei gewählt werden. Alle diese Suchkriterien lassen sich frei kombinieren, wodurch sich z.T. sehr differenzierte Fragen beantworten lassen.

So sucht der Computer in Sekundenschnelle alle Patienten heraus, die auf der Intensivstation im 1. Quartal wegen einer Lungenentzündung beatmet werden mußten, älter als 60 Jahre und männlich waren. Daraus kann man dann ersehen, mit welchen Bakterien bei diesen Patienten am ehesten zu rechnen ist und welche Antibiotika am besten helfen. Dies kann manchmal einen lebensrettenden Zeitgewinn bedeuten, bis das Ergebnis der Untersuchungen für einen einzelnen Patienten nach Tagen vorliegt! Wenn dann diese Rasterung für fortlaufende Quartale durchgeführt wird (was das Programm komfortabel unterstützt), kann man auch zeitliche Trends erkennen. So kann man rechtzeitig etwas unternehmen, wenn ein Bakterium überhandnimmt oder ein Antibiotikum immer schlechter wirkt.

(Fast) alles ist möglich

Natürlich kann man auch ziemlich unsinnige Rasterkombinationen vorgeben. Dann bekommt man ein zwar korrektes, aber wenig sinnvolles Ergebnis. Und wie mit jeder Statiskik läßt sich auch damit viel Unsinn treiben. So stellt auch weiterhin der Arzt die Diagnose bzw. entscheidet sich für ein Antibiotikum, und nicht unser Atari, auch wenn wir ihm sonst so viel vertrauen.

Das Programm läuft nun schon seit ca. 18 Monaten und hat uns manches interessante Ergebnis geliefert. In dieser Zeit wurden wiederholt Änderungen daran vorgenommen. Insbesondere haben wir weitere Funktionen eingebaut, wenn uns diese wichtig und machbar erschienen. So hat es unterdessen einen gewissen Reifegrad erreicht. Seine Funktionen entsprechen den praktischen Anforderungen und nicht nur theoretischen Überlegungen. Seit kurzem ist auch die Anpassung an die Gegebenheiten anderer Krankenhäuser bzw. bei Änderungen im eigenen Haus leicht mit Hilfe eines weiteren kleinen Zusatzprogrammes möglich.

Es geht nicht ganz ohne Probleme

Bei aller Begeisterung bleiben jedoch auch Probleme. Das größte besteht im hohen Aufwand an Tipparbeit bei der Eingabe in Anbetracht von ca. 10000 Befunden pro Jahr. Deshalb wurde darauf geachtet, daß wirklich nur ein Minimum an Tastendrücken pro Befund nötig ist. So dauert die Eingabe eines Befundes für den etwas eingearbeiteten Nutzer nur wenige Sekunden. Auch die beim ST-Computer so wichtige Maus kommt im Auswertungsprogramm voll zu Ehren, sie hilft bei der Auswahl aus den Menüs, alternativ kann aber meist auch eine Taste gedrückt werden. Und so richtig austoben kann sich jeder Nagerliebhaber im zugehörigen Grafikprogramm, denn das wurde voll unter GEM geschrieben. Schön wäre jedoch die Dateneingabe über strichcodemarkierte Laborzettel. Doch dann ergäbe sich das neue Problem, ein Strich-Code-Lesegerät per Programm abzufragen. Dazu fehlen uns die Erfahrungen. Außerdem müßten die Diagnosen weiterhin per Hand eingetippt werden, da es sich ja um frei vom Benutzer gewählte Begriffe und nicht nur um eine Auswahl aus einer kleinen Liste handelt.

So wird denn trotz hervorragender Unterstützung durch den ST-Computer weiterhin eine Menge Fleiß nötig sein, bevor der Preis der Erkenntnis aus der Rasterfahndung winkt. Und noch etwas bleibt: Gegen Computerviren, die sich an unserem Atari zu schaffen machen, hilft das Programm leider auch nicht!

Dr.med. Manfred Kester
Am Römerkastell 1
W-6350 Bad Nauheim

nicht zugeführt. „Subliminal" heißt insgesamt der Vorgang der unbewußten Kommunikation.

Seit geraumer Zeit ist eine andere Spielart der unterschwelligen Informationsübertragung zu beobachten, das sogenannte 'Lernen im Schlaf', oder neumodischer ausgedrückt: Superlearning. Es gibt zwei völlig verschiedene Verfahren des Superlearnings. Einerseits wird durch Sphärenklänge ein Beruhigungszustand erreicht, bevor beispielsweise ein Fremdsprachenprogramm abläuft. Die andere Technik bedient sich bestimmter Vorgänge beim nächtlichen Tiefschlaf.

Zur Zeit geht in den Vereinigten Staaten die Diskussion um eine ähnliche Einflußnahme an Computerarbeitsplätzen. Es gibt die Bestrebung, den Arbeitern und Sekretärinnen an den Terminals unterschwellig solche Meldungen wie „Ich liebe meine Arbeit.", „Ich arbeite gerne hier.", „Ich freue mich auf mein Büro." usw. unterzujubeln. Es sind bereits mehrere Verfahren anhängig, die eine derartige Manipulation im Sinne der Arbeitgeber zu unterbinden versuchen.

Subliminale Texte

Im Grunde ist es technisch überhaupt kein Problem, solche unterschwelligen Botschaften per Bildschirm zu übermitteln. Vor kurzem wurde uns ein Programm vorgelegt, das nunmehr auch auf dem Atari ST/TT solche subliminalen Mitteilungen erzeugt. Sein Name: MIND_SUP.ACC (was immer das auch heißen mag). Die Firma Olaf Blum in Berlin stellt uns diese Software, die als Accessory betrieben wird, mit einem sogenannten Gesundheitsmodul vor.

Eigentlich hätte es ein Programmtest werden sollen. Nachdem ich mich aber etwas genauer mit Subliminals im allgemeinen und mit der Wirkung des Programms im speziellen beschäftigt habe, wollten wir eine Besprechung dieses Produktes doch unter einer anderen Rubrik erscheinen lassen - so wurde ein ST-Report daraus. Im Grunde hätte ein Test auch nicht viel erbracht. Das Programm lauert im Hintergrund, blendet kurzzeitig diverse Schrifttafeln ein - und das war's denn auch schon.

Ob irgendwann einmal Resultate aus den subliminal hervorgebrachten Texten festzustellen sind, müßte über einen langen Zeitraum beobachtet werden. Und dann wäre nicht auszuschließen, daß es mir wohl ohne subliminales Programm auf meinem ST kaum anders (besser oder schlechter) ergangen wäre. Daß das Programm funktioniert, will ich gerne bestätigen.

Lassen Sie uns dennoch einen kleinen Blick auf das Programm selbst werfen, das als ACC installiert sein soll. Zu dem Rumpf-ACC gehören verschiedene Module, die entsprechende Texte zu fest umrissenen Wissensgebieten enthalten. Uns wurde ein Gesundheitsmodul beigelegt. Die Texte befassen sich allesamt mit gesundheitlichem Wohlbefinden und körperlicher Reinheit.

Beispiele: „Meine Sinne arbeiten für mich und sorgen für einen optimalen Austausch mit meiner Umwelt." „Der Leistungsumfang meines Hirns wächst ständig und weitet mein Bewußtsein Tag für Tag."

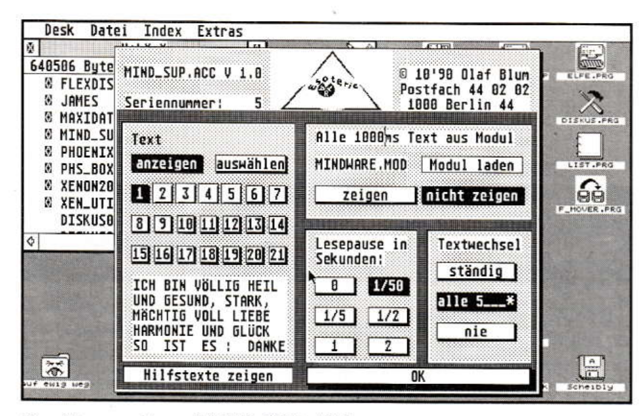
Als Informatiker bin ich Realist, ich habe im Studium gelernt, wie man analytisch auswertet, wie man nach fest vorgegebenen Formeln Gleichungen löst. Ich glaube auch, daß es Techniken gibt, die an das Unterbewußtsein gelangen und ihm Informationen weitergeben, ohne daß ich es wahrnehme. Aber ist das alles so einfach?

Reicht es einfach aus, kurzzeitig Schrifttafeln auf meinen Bildschirm zu blitzen - und schon fühle ich mich besser? Oder genügt ein subliminaler Hinweis, während ich mit der Textverarbeitung wirke - und schon ist mein Heißhunger nach Gummibärchen gestillt? Ich halte es mit Shakespeare: „.... allein mir fehlt der Glaube."

Das Programm kostet DM 199,- inkl. Gesundheitsmodul, alle weiteren Textmodule gibt es zum Preis von DM 99,-. Sonderwünsche bezüglich der Texte ist man bereit, für DM 200,- pro Modul zu erfüllen. Angesichts der schweren Nachprüfbarkeit einer Wirkung erscheint mir die Preisgestaltung um einige „Etagen" zu hoch. Zudem befaßt sich das 13seitige Handbüchlein nur sehr oberflächlich mit der zugrundeliegenden Technik der Subliminalmethode.

Wie schon gesagt, es hätte eigentlich ein Programmtest werden sollen. Die Rubrik „ST-Report" befaßt sich vornehmlich mit außergewöhnlichen Anwendungen auf oder mit dem Atari-Computer - und dieses Subliminalprogramm gehört durchaus dazu. In meinen Augen sind die gerade erst in Erforschtung begriffene Unterbewußtseinsebene unseres Gehirns und die dortigen Vorgänge ungemein interessant, andererseits aber sind uns Anwendungen wie das Programm MIND_SUP.ACC den Beweis noch schuldig, ob sie wie gewünscht und erwartet funktionieren. Ihre reine Existenz und Ablauffähigkeit beweisen das noch nicht.

DK



Das Steuerpult von MIND_SUP.ACC

ICH BIN VÖLLIG HEIL UND GESUND, STARK, MÄCHTIG VOLL LIEBE HARMONIE UND GLÜCK SO IST ES : DANKE

DER LEISTUNGSUMFANG MEINES HIRNS WÄCHST STÄNDIG UND WEITET MEIN BEWUßTSEIN VON TAG ZU TAG ! DANKE !

ICH BIN EIN STRAHLENDES WESEN, AUF DEM WEG, DIE VERANTWORTUNG MEINER GÖTTLICHKEIT ANZUNEHMEN!

MEINE AUGEN SIND MIR GUTE WERKZEUGE ! SIE SEHEN GUT, SCHARF, UND IN ALLEN BEREICHEN ! DANKE !

MEIN SCHÖNER KÖRPER IST MEIN GESUNDER HELFER. MEINE AUFMERKSAMKEIT SCHÜTZT IHN JEDERZEIT! DANKE

MEINE NASE IST MIR EIN GUTES WERKZEUG ! SIE FILTERT FREMD - STOFFE UND RIECHT WICHTIGES ! DANKE !

Eine kleine Auswahl subliminaler Texte

SCSI-Festplatten zu »Schotten-Preisen«!



Unsere SCSI-Festplatten werden komplett anschlußfertig incl. Software und Kabel ausgeliefert.

Ausstattung und Leistungsmerkmale unserer Festplatten:

- Datentransferraten >600KByte/s (mit CDC- und Maxtorlaufwerken bis zu 1300 KByte/s erzielbar), mittlere Zugriffszeiten bis zu 14 ms
- Spitzensoftware: 255 Partitionen installierbar, Passwortfunktion, jede Partition autobootfähig, Interleave 1:1 einstellbar, Cache, Backup, Optimizer in der Software enthalten
- 100% Atari-kompatibel, sämtliche Fremdbetriebssysteme (PC-Speed, PC-Ditto, Spectre, Aladin, Minix, OS-9, RTOS) sind voll lauffähig
- Superleise (3,5"-Festplatten ohne Lüfter, 5,25"-Festplatten mit thermogeregeltem Lüfter)
- Durchgeschleifter gepufferter DMA-Bus, Autoparkfunktion hardwaremäßig
- Herausgeführter SCSI-Bus (50-poliger Centronics-Anschluß, Apple MacIntosh und PC's anschließbar)
- Zweite SCSI-Festplatte im Gehäuse nachrüstbar (SCSI-Hostadapter und Gehäuse für interne zweite Festplatte vorbereitet)
- Unsere SCSI-Festplatten werden komplett anschlußfertig im Gehäuse incl. Netz-, DMA-Kabel, Software und Handbuch geliefert

Preise:

32 MB, 40 ms, ST138N-0	DM	998,–
40 MB, 19 ms, Quantum	DM	898,–
49 MB, 28 ms, ST157N-1	DM	1.098,–
85 MB, 28 ms, ST296N	DM	1.198,–
80 MB, 24 ms, ST1096N	DM	1.298,–
105 MB, 19 ms, Quantum	DM	1.398,–
170 MB, 28 ms, 2xST296N	DM	2.498,–
280 MB, 17 ms, Maxtor	DM	3.498,–
380 MB, 17 ms, Maxtor	DM	3.998,–
702 MB, 14 ms, CDC	DM	5.998,–
1200 MB, 14 ms, CDC	DM	11.998,–
44 MB, 25 ms, SQ 555	DM	1.398,–
SCSI-Kits (Festplatte und SCSI-Hostadapter für ST):		
32 MB Kit (ST138N-0)	DM	798,–
40 MB Kit (P40S)	DM	798,–
49 MB Kit (ST157N-1)	DM	898,–
85 MB Kit (ST296N)	DM	998,–
80 MB Kit (ST1096N)	DM	1.098,–
105 MB Kit (P105S)	DM	1.198,–
SCSI-Hostadapter (incl. Software und DMA-Kabel	DM	198,–
DMA-Kabel	DM	39,–
SCSI-Kabel	DM	39,–
Netzteil 50 W	DM	99,–
Gehäuse	DM	99,–
Cartridge für SQ555	DM	198,–

Weitere Modelle sowie sonstige Soft- und Hardware auf Anfrage!



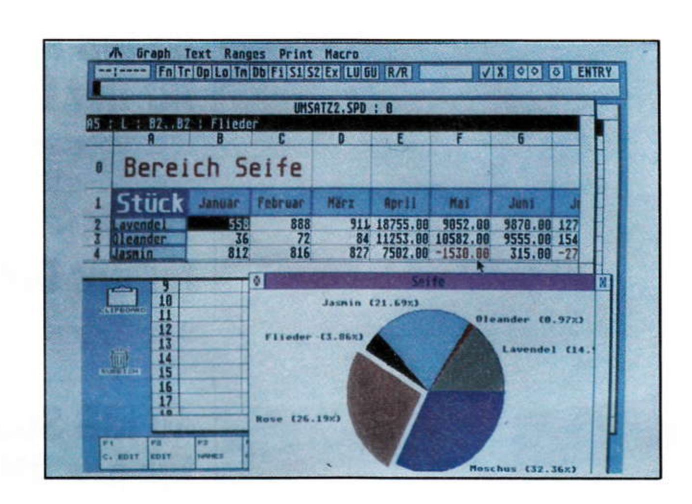
23. bis 25. August 1991



Datensysteme

Eugenstraße 28
7302 Ostfildern 4
Telefon 0711/4579623
Telefax 0711/4569566

K-Spread 4



Das Leben in der Zelle

Sagte doch Heinz Erhard: „Das Leben beginnt, auf alle Fälle, in einer Zelle und endet oft, bei Strolchen, in einer solchen." Und wenn wir es genau nehmen, findet das „Leben" einer Tabellenkalkulation auch in Zellen statt. Nun stellt sich uns die Frage: „Wat is egentlich ene Tabellenkalkulation?" Worauf die Antwort folgt: „Dat isene Programm, da kann man wat reintun, nämlich Daten." (Zitate aus dem Handbuch - geistiger Vater unbekannt).

Diese besagten Daten nun aber, es kann beispielsweise ein Text, eine Zahl oder eine Formel sein, schreibt man nicht wahllos auf den Bildschirm, sondern sie werden von jenen Zellen aufgenommen, die das gesamte Rechenblatt ausfüllen. Zellen sind also eine Art Speicherfächer innerhalb dieses Programms. Apropos: Computerprofis benutzen in diesem Zusammenhang gerne auch den aus der Datenbankterminologie entlehnten Begriff „Felder", der natürlich ebenso seine Gültigkeit hat.

Wozu braucht man denn nun typischerweise ein Tabellenkalkulationsprogramm? Um vielleicht gerade einmal kurz einige Zahlen zu addieren, dafür gibt es bekanntermaßen Taschenrechner beliebiger Bauart, auch Kopfrechnen soll wieder in Mode kommen. Der gefürchtete Dreisatz dürfte mit ein wenig Papier und Bleistift schneller gelöst sein, als wenn ein Großcomputer hierzu beansprucht werden würde. Für alles das lohnt es sicher nicht, den Computer nebst Festplatte warmlaufen zu lassen.

Andererseits gibt es ganz spezielle Rechenbeispiele, die auch schon die Grenze eines Tabellenprogramms sprengen. Obwohl solche Software durchaus mit viel Zeit und Speicherplatz in der Lage wäre, eine Industriebilanz oder eine umfangreiche Konzernbuchhaltung zu bewältigen.

Wie dem auch sei, man kann sich durchaus eine Vielzahl von Anwendungen vorstellen, bei denen eine Tabelle genau die richtige Arbeitsplattform ist. Gerade wenn es darum geht, regelmäßig gleichverlaufende Rechenarbeit zu delegieren, also z.B. die Gehaltsabrechnung mit ihren verschiedenen Abhängigkeiten zu Sozialabgaben und Steuern, den Lohnsteuerjahresausgleich mit den verschiedensten Höchst- und Mindestsätzen oder gar eine Auftragsabwicklung durchzurechnen (halt wirklich zu „kalkulieren"), dann wird die „Tabelle" wahrscheinlich genau das richtige sein.

Bild 1: Das Antlitz von K-Spread: typisch Tabelle

Alte Namen - neue Namen

Kaum eine andere Branche ist so schnellebig, wie die der Computer und ihrer Anwendungen (wem sag ich das). So gab es auch in der Atari-Welt klangvolle Titel von Tabellenprogrammen, wie VIP professional, Logistix, LDW-Powercalc, Basicalc. Mit ein Programm der ersten Stunde war K-Spread von der englischen Firma KUMA. (Übrigens gesprochen wird K-Spread: „Käsbrett".) Nunmehr legt uns

```
Datei
  Rechenblatt laden..        ^L
  Rechenblatt sichern..      ^S
  Bereich speichern..
  Bereich einladen..         ^A
  ASCII-Text laden..
  ------------------------------
  Passwort..
  ------------------------------
  Export..                   ^G
  Import..                   ^T
  ------------------------------
  Neue Funktionen laden..
  Neue Funktionen sichern
  ------------------------------
  Druckereinstell. laden
  Druckausgabe in Datei
  ------------------------------
  Datei löschen..
  ------------------------------
  K-Spread verlassen         ^Q

Arbeiten
  Datenblatt anlegen..       ⌘A
    umbenennen..             ⌘N
    entfernen..              ⌘Z
  ------------------------------
  Suche/Ersetze..            ^F
  Gehe zu Feld..             ⌘G
  Sortiere..                 ⌘S
  Neue Funktionen..          ^U
  ------------------------------
  Bereich füllen..           ^I
  Bereich kippen..           ^J
  ------------------------------
  Eingabebereich 1..         ⌘1
  Eingabebereich 2..         ⌘2
  Häufigkeitsverteil.        ⌘D
  ------------------------------
  Matrizen multipliz.        ^Y
  Matrix invertieren..       ⌘Y
  ------------------------------
  Bereich aufspalten..
  Formeltyp ändern..         ⌘C

Blatt
  Zahlen-Format..            ⌘V
  Formel-Format..            ⌘F
  Text-Format..              ⌘L
  Prefif/Suffix..            ⌘B
  Format festlegen..         ⌘P
  ------------------------------
  Neuberechnen festl.        ^R
  Allg. Format festl.        ⌘J
  ------------------------------
  Felder überwachen..        ⌘T
  Feldanfangswerte..         ⌘I
  ------------------------------
  Bereichsnamen..            ⌘R
  ------------------------------
  Spaltenbreite..            ^W
  Zeilenhöhe..               ^H
  ------------------------------
  Datenblatt löschen..       ^Z
  ------------------------------
  Kopieren..                 ^C
  Verschieben..              ^M
  Löschen..                  ^D

Fenster
  öffnen..                   ⌘O
  Fenstergröße..             ⌘W
  ------------------------------
  Zeige Formeln
  Zeige Datentypen
  Null nicht anzeigen
  ------------------------------
  Zeichensatz & Größe..      ^V
✓ Gitter
  ------------------------------
  Spalte festhalten..        ^K
  Zeile festhalten..         ⌘K
  ------------------------------
  Zum nächsten Fehler
  ------------------------------
  Umranden..
  Graph..

Datenbank
  Datenbereich..
  Kriterienbereich..
  Ausgabebereich..
  ------------------------------
  Suche
    alle ausgeben
         ohne Duplikate
    alle löschen
  ------------------------------
  Datensatz einfügen
  Datensatz löschen
  ------------------------------
  Datenbank schließen

Drucken
  Format festlegen..
  Druckereinstellung..
  Steuerzeichen..
  Kopf-/Fußzeilen..          ⌘H
  ------------------------------
  Drucke..                   ^P
    mit GDOS..
    Inhalte..
  ------------------------------
  Seitenvorschub
  Zeilenvorschub
  Seitenanfang festlegen

Global
  Warnmeldungen..
  ------------------------------
  Umgebung sichern
  Umgebung aufräumen
  ------------------------------
  Pop Up Menüs               ^E
  ------------------------------
  Neustart (!)
  ------------------------------
  Speicher..

Macro
  Macro anlegen..
  Macro umbenennen..
  Macro löschen..
  Macroanfang setzen
  ------------------------------
  Macrotaste..
  ------------------------------
  immer Neuzeichnen
  Macro ausführen..
  Timer..
  ------------------------------
  Aufnahme starten
```

Bild 2: In der Hauptmenüleiste ist es wegen Befehlsgedränge schon fast zu eng geworden.

die Firma OMIKRON aus Pforzheim „K-Spread 4“ vor, wobei der Name signalisiert, daß es sich hierbei um eine Weiterentwicklung des bekannten KUMA-Produktes handelt.

Auf den ersten Blick erkennt man durchaus Ähnlichkeit im Aussehen der Arbeitsoberfläche mit dem der Mitbewerber (man möge mir diese Bemerkung verzeihen). Da haben wir die typische Anordnung der Zellen in einem Fenster, Zeilen haben Kennummern, Spalten haben Kennbuchstaben, sehen oben unsere Menüzeile, auch einige Icons zieren das Bild. Aber bald ist es mit der Ähnlichkeit vorbei.

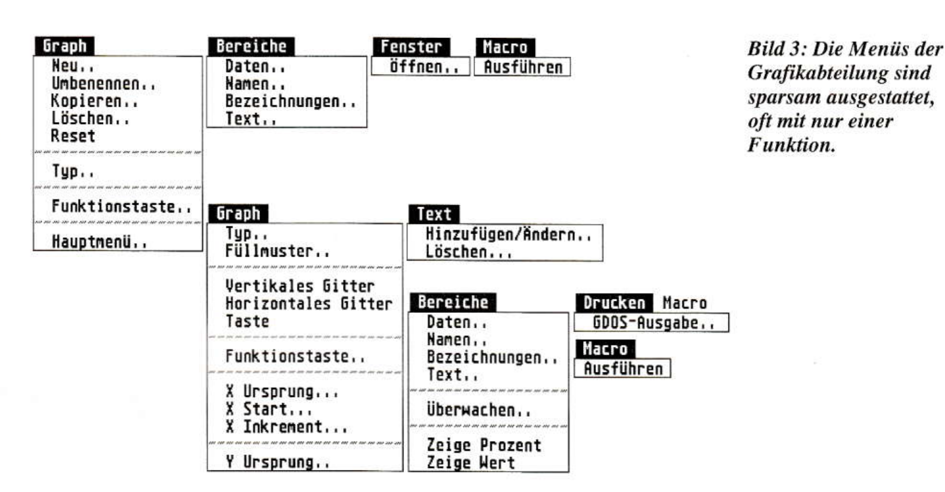


Bild 3: Die Menüs der Grafikabteilung sind sparsam ausgestattet, oft mit nur einer Funktion.

Globale „innere“ Werte

Sehen wir uns doch gleich einmal die technischen Daten von K-Spread an: Pro Arbeitsblatt sind bis zu 8192 Zeilen mal 256 Spalten, also 2097152 Zellen definiert. „Definiert“ heißt, es gibt so viele Bezeichnungen für Zeilen und Spalten, das bedeutet aber noch lange nicht, daß diese auch bis zum Rand mit Daten gefüllt werden können. Gerade das ist das Hauptproblem aller Tabellenprogramme, daß sie uns ein astronomisch großes Arbeitsblatt vortäuschen, das überhaupt nicht völlig ausgenutzt werden kann. Sehr oft macht uns eine physische Grenze im Computer einen Strich durch die Rechnung: der freie Arbeitsspeicher.

Testen wir doch K-Spread einmal: Am besten eignet sich hierzu die sogenannte Fill-Funktion. Mein Mega ST2 meldete mir 1619 kByte freien Arbeitsspeicher, wovon 370 kByte durch K-Spread und weitere 16 kByte durch das offene, noch leere Arbeitsfenster beansprucht wurden. Dann ließ ich die Fill-Funktion einfach die Zellen „A0“ bis „Z5000“ mit fortlaufenden Zahlen (Iteration 1) vollschreiben, was eine Gesamtanzahl von 130025 Zellen betraf. Das Programm benötigte dafür 4 Minuten 35 Sekunden und meldete mir hernach „Out of Memory“ (was eigentlich zu erwarten war). Fast wäre also die gewünschte Operation vollständig abgelaufen, denn es wurden effektiv 109785 Zellen vollgeschrieben.

Hierdurch sieht man sehr deutlich, daß der sogenannte nutzbare Tabellenraum in Wahrheit viel kleiner ausfällt als der theoretisch definierbare (siehe oben). Aber dafür kann K-Spread nun wirklich nichts, das ist eine typische „Berufskrankheit“ aller Tabellenprogramme. Letztenendes fallen die meisten Tabellenanwendungen nicht so umfangreich aus, daß in der Tat mehr als 100000 ansprechbare Zellen (bei meinem Mega ST2) oder gar über 2 Millionen (theoretisch) wirklich nötig sind.

Auch am Arbeitsspeicher orientiert sich ein weiteres Kriterium: K-Spread erlaubt, übrigens als einzige Tabellenkalkulation auf dem ST/TT, beliebig viele Arbeitsblätter offenzuhalten, womit wohl mehr das gleichzeitige Offenhalten von Datendateien gemeint ist. Darstellbar sind dann maximal 8 gleichzeitig offene Fenster. Eine Funktion des Programms nutzt dies nun aus. Es ist eine „übergreifende Cut-and-Paste“. So kann ein markierter Block in einem Rechenblatt unmittelbar in ein anderes (offenes) übertragen werden. Der Umweg über ein Klemmbrett oder gar ein Export-Import wie bei anderen Programmen ist nicht mehr nötig. Abgesehen davon ist auch ein Zugriff auf Werte oder Inhalte anderer (ungeöffnete) Tabellen, z.B. aus Formeln heraus, problemlos möglich.

Das Menü bitte

Also einmal einen Sprung in die Menüleiste gewagt (Bild 2). Das sind nicht gerade wenig Einträge, die ihre Funktionen dort verstecken. Trotz der Fülle sind die Pull-Down-Menüs den GEM-Restriktionen gemäß geordnet, beispielsweise durch Trennstriche abgegrenzt. Sehr sinnvoll ist die Anmerkung bei den meisten Menüeinträgen, daß die dortigen Arbeiten auch per ALT- oder CTRL-Tastenfunktion startbar sind. (Früher oder später kehren Sie ohnehin zu den Tastaturkürzeln zurück.) Wenn wir dann noch in den Grafikmodus schalten oder ein Grafikfenster aktivieren, tut sich uns eine weitere Menüleiste auf, die nunmehr Grafikfunktionen offenbart. Diese Praxis zeigt, daß den Programmierern die normale Menüleiste schon zu voll geworden ist (ein altes Lied).

Mehr durch Zufall (weil ich das Handbuch nicht lese) bin ich auf eine zweite, direkt zugängliche Menüleiste (Fn Tr Op Lo Tm Db Fi S1 S2 Ex) gestoßen. Sie „verbirgt“ sich unmittelbar unter der obersten Hauptleiste und heißt Funktionenmenü. Hier ist nun aber die Bezeichnung mit den „Funktionen“ wirklich angebracht, denn es sind dort eben die über 100 mathematischen, logischen, Finanz-, String- und Datenbankoperationen untergebracht (siehe Bild 3). Bemerkenswert: Es können beliebig viele weitere Funktionen bzw. Operationen vom Benutzer frei definiert und abgespeichert werden.

Und nun gibt es auch noch am untersten Bildrand eine Funktionstastenleiste. Dort sollen laut Handbuch die am häufigsten benutzten Aufgaben zu finden sein. Nun, was „Ändern“ (F1), „Eingeben“ (F2) und

Funktionen	Trigon.	Op's	logische Op	Zeitfkt.	Datenbank	Finanzfkt	Textfkt.	Textfkt.	Funktionen
ASKN	ABS	+	IF	DATE	DAVG	CTERM	CHAR	EVALUATE	ATOCOL
ASKT	ACOS	-	THEN	DATEVALUE	DMEAN	DDB	CLEAN	REVERSE	COL
AVG	ALOG	*	ELSE	TIME	DCOUNT	FV	CODE	ROTATE	COLTOA
CHOOSE	ASIN	/	<	TIMEVALUE	DMAX	IRR	EXACT	STRCNT	CONTENT
COLS	ATAN	:	<=	DAY	DMIN	NPV	FIND	STRDAY	CONVERT
COUNT	ATAN2	^	>	WEEKDAY	DSTD	PMT	LEFT	STRDEL	DAYSINYEAR
ERR	COS	(	>=	MONTH	DSTDP	PV	LENGTH	STRDELM	DAYSINMONTH
HLOOKUP	EXP	)	=	YEAR	DSUM	RATE	LOWER	STRJUSTIFY	DIFF
INDEX	INT	[	<>	FYEAR	DVAR	SLN	MID	STRMAP	FILE
MAX	LN	]	AND	HOUR	DVARP	SYD	N	STRMONTH	HEIGHT
MEAN	LOG	,	OR	MINUTE		TERM	PROPER	STRSCAN	ISLEAPYEAR
MIN	MOD	~	NOT	SECOND			REPEAT	STRNSCAN	POWER
NA	PI	..	TRUE	NOW			REPLACE		ROOT
ROWS	RAND	$	FALSE	DTOS			RIGHT		ROW
STD	ROUND	@	ISERR	HTOS			S		SUMNEG
STDP	SIN		ISNA	MTOS			STRING		SUMPOS
SUM	SQRT		ISNUMBER	STOD			TRIM		WIDTH
VAR	TAN		ISSTRING	STOH			UPPER		
VARP	TRUNC		ISEMPTY	STOM			VALUE		
VLOOKUP									

Bild 4: Die Auswahl an Operatoren

„Gehe zu“ (F5) anbelangt, glaub' ich das gerne, aber kommt es denn so oft vor, daß man ständig zwischen globaler Neuberechnung des gesamten Arbeitsblattes (F9) und Neuberechnung eines definierten Bereiches (F10) wählen muß? Da hätte man doch sicher weit sinnvollere Tätigkeiten finden können, die einer Beförderung zur Funktionstaste wert gewesen wären. Mein Urteil: Noch einmal darüber nachdenken.

Links und rechts vom Funktionenmenü sind noch einige Anzeigeelemente und winzige „Schalterchen“ zu sehen, die uns im Moment (weil leeres Rechenblatt) nicht viel zu sagen haben. Ganz schnell ins Auge fällt - die Edit-Zeile, weil sie sich über die ganze Bildschirmbreite erstreckt. Sie reflektiert eine bevorstehende Eingabe, wenn man gerade schreibt oder einen wahren Inhalt, wenn der Cursor auf einer belegten Zelle steht.

Als Fazit zu diesem Kapitel läßt sich durchaus ein kleiner Minuspunkt vermerken: K-Spread hat verwirrend viele Bedienungselemente, was andere Rezensenten (je nach Standpunkt des Betrachters) auch wieder positiv auslegen können: „Das Programm kann eben viel“. Dennoch stoßen die Riesenauswahl in den Pull-Down-Menüs, die zweite Menüleiste sowie eine zusätzliche Ausnutzung von Funktionstasten, Icons und Schaltern in meinen Augen an die Grenze der Übersichtlichkeit. Es liegt der Verdacht nahe, daß neu hinzugekommene Programmfunktionen einfach zusätzlich in die Menüs verfrachtet wurden, so wie zwangsweise neue Bedienungselemente hinzuerfunden werden mußten.

Hinein ihr Daten

Dort, wo sich eine Tabellenzeile und eine Tabellenspalte treffen, liegt die obligatorische Zelle. Mit dem Cursor (hier sehr vornehm „Zellzeiger“ oder „Feldzeiger“ genannt) steuert man im Rechenblatt eine bestimmte Zelle an. Die Auserwählte darf sich jetzt „aktuelle Zelle“ nennen, denn alles, was nun über die Tastatur Eingang findet, wird nur in diese Zelle geleitet. Es gibt also immer nur eine aktuelle Zelle, wie bei einem riesigen Essensspender im Automatenrestaurant, wo nach Münzeinwurf auch nur ein Klappfensterchen geöffnet werden kann.

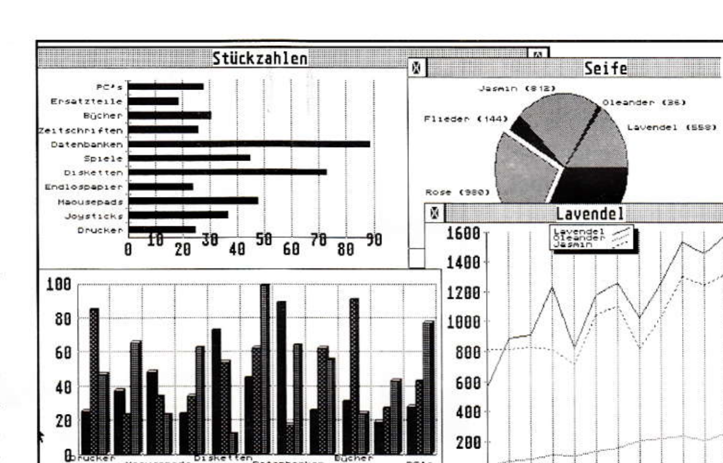


Bild 5: Ein kleine Auswahl der Grafikmöglichkeiten

Apropos Cursor: Sie haben zwei verschiedene auf dem Bildschirm! Schreibmarke Nr. 1 (auch Edit-Cursor genannt) gibt sich in weißer Farbe und ist zur besseren Sichtbarkeit als kleiner Rahmen dargestellt. Er verdient den Namen zurecht, denn er ist für Eingaben in die aktuelle Zelle vorgesehen und wird ausschließlich mit den Cursor-Steuertasten bedient. Zellzeiger Nr. 2 wird sinnvollerweise auch Markierer genannt und bleibt in schwarzer Farbe auf dem Bildschirm. Ihm ist die Aufgabe zugedacht, u.a. für Markierungen von Bereichen, Bewegungen im Arbeitsblatt oder Zellkopien dazusein und wird ausschließlich mit der Maus bedient. Das Hantieren mit zwei verschiedenen Cursor'n (oder heißt das „Cursors“?) ist anfangs stark gewöhnungsbedürftig, wohl auch deshalb, weil es so etwas in anderen Tabellenprogrammen nicht gibt. Dennoch finde ich diese Idee genial.

Und hier gleich zu weiteren drei Pluspunkten von K-Spread:

1. Die Feldbreite darf bis zu 80 Zeichen groß werden (mehr geht in normaler Systemschrift ohnehin nicht auf den Schirm). 2. Es gibt eine sogenannte Feldhöhe(!), die bis zu 14 Zeilen umfassen darf. Genau dieser Umstand ist eines der herausragenden Merkmale von K-Spread, denn bei keiner anderen Tabellenkalkulation gibt es mehrzeilige Felder. 3. Innerhalb eines Mehrzeilen-Textfeldes gibt es eine Wortumbruchfunktion (engl. wordwrap). Damit ist eine ausreichend nützliche Textverarbeitung möglich.

Tabellenprofis wissen es schon, es gibt drei verschiedene Datentypen, die unser Programm akzeptiert: Text, Zahl oder Formel. Sehr schön ist in diesem Zusammenhang eine Automatik, die sofort nach der ersten Zeicheneingabe feststellt, um welchen Datentyp es sich handelt (Kunststück). Wenn also eine Buchstabentaste als erstes gedrückt wird, liegt für diese Zelle der Datentyp Text fest. Eine Zelle kann bis zu 128 Zeichen aufnehmen. Abweichend von den Typen Text und Zahl bringt die Formel nicht zwangsläufig auch ihren wahren Inhalt (also die geschriebene Formel selber) auf den Bildschirm, sondern das schon errechnete Ergebnis. K-Spread 4 kennt drei verschiedene Arten von Operatoren, die übrigens durchaus bunt gemischt in einer Formel vertreten sein dürfen: arithmetische, logische und Textoperatoren.

Absolut ist relativ

Jede Zelle trägt die Kennzeichnungen der sie kreuzenden Zeile (eine Zahl) und Spalte (Buchstaben). Diese Angabe lokalisiert eine einzelne Zelle eindeutig gegenüber anderen, weshalb man diese Koordinatenangabe auch „Lokalisator" nennt. Nun kann es in einer Formel (durchaus typisch und nicht gerade selten) vorkommen, daß mit der Koordinaten einer anderen auf deren Inhalt Bezug genommen wird. Will heißen: Man braucht deren Inhalt zum Weiterrechnen. Also wird einfach die Koordinate eingetragen.

Jetzt kommt aber der Clou: Es gibt zwei verschiedene Typen von Koordinaten: relative und absolute. Der Unterschied macht sich beim Kopieren oder Verschieben der Zelle (mit der Formel drin) bemerkbar. Eine absolute Koordinate in einer Formel, z.B. „D9" bezieht sich immer und jederzeit auf diese Zelle D9, gleichgültig wo sie im Arbeitsfenster sonst noch hinkopiert wurde. Eine relative Kennzeichnung verändert sich, wenn sich ihre Lage verändert. Wenn man also die Formel mit der „D9", um drei Zeilen nach unten verschiebt, ändert sich die „9" in eine „12". Beim Kopieren um eine Spalte nach rechts wird aus dem „D" ein „E" werden. Der sogenannte Bezugsabstand hat sich verändert, und die relative Angabe in der Formel ändert sich entsprechend.

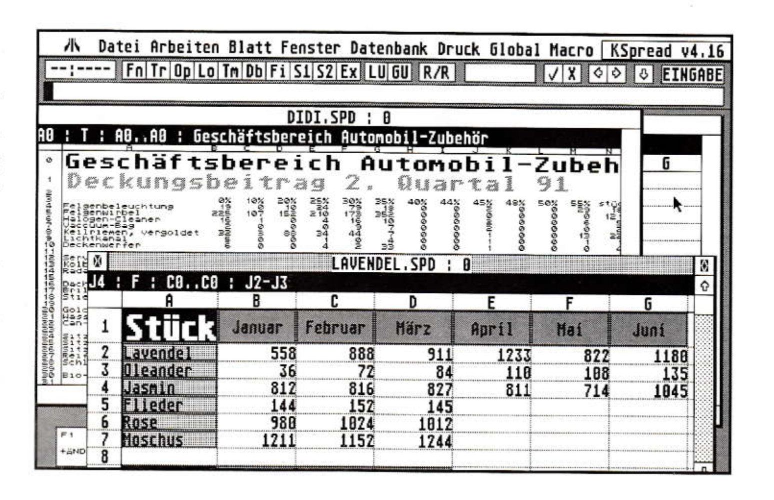


Bild 6: Hier sieht man die Wirkung der verschieden großen Zellen besonders gut.

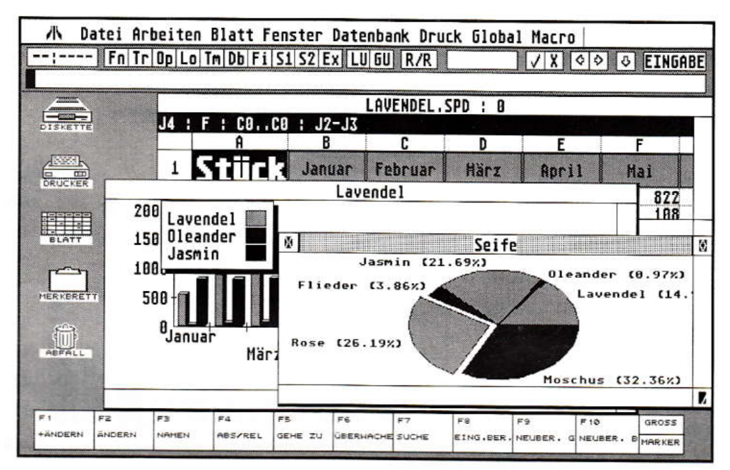


Bild 7: Das Ziel der Arbeit: sinnvolle Grafik, komplett mit automatischer Beschriftung

Datenbank

Da sich bei einer Tabellenkalkulation ohnehin alles in geordneten Kolonnen abspielt, lag der Schritt nahe, neben reinen Rechenaufgaben auch Datenbankfunktionen zu bescheren. So ist es auch problemlos möglich, die Tabelle als Datenbank, allerdings nur in der Listenform, zu verwenden. Besonders die geschickte Kombination von Datenbankbereich und Tabellen- bzw. Rechenbereich läßt Raum für sinnvolle Eigenentwicklungen.

In K-Spread besteht ein Datensatz grundsätzlich aus einer Tabellenzeile, die einzelnen Datenfelder sind die Zellen. Die oberste Zeile nimmt dabei eine Sonderstellung ein, sie gibt die maßgeblichen Feldnamen vor. Um nun beispielsweise nach einem bestimmten Namen suchen zu lassen, muß vorher fernab vom Datenbereich ein Kriteriumsbereich definiert werden. Dort müssen sowohl der Feldname als auch das Suchkriterium (also der gesuchte Name selber) explizit untereinander in einen freien Bereich der Tabelle geschrieben sein. Als Suchbegriff sind natürlich nicht nur Zahlen und Text zugelassen, sondern auch eine Formel, die beispielsweise eine logische Bedingung sein kann. In solchen Formeln sind alle mathematischen (größer, kleiner ...) und logischen Operatoren (alles, nichts, oder ...) zugelassen, die auch gemischt und verschachtelt definiert sein können.

Des weiteren gibt es spezielle Datenbankfunktionen, die auf einen fest vorbestimmten Datenbereich oder die ganze Tabelle anwendbar sind: Durchschnittsberechnung, Anzahl der Datensätze, größter oder kleinster Wert einer Datenreihe bis hin zur Standardabweichung (um nur einige zu nennen).

Grafik

Jetzt müßte der Standardspruch wieder kommen: „Ein Bild sagt mehr ..." Natürlich ist es sinnvoll, sich Ergebnisse langer Kolonnen, Vergleiche verschiedener Resultate oder einfach nur Gegenüberstellungen ähnlicher Sachverhalte grafisch ausgeben zu lassen. So gibt es wahlweise Torten-, 3 verschiedene Balken- und 2 Liniengrafiken, die aus dem Zahlenmaterial ansehnliche Diagramme fabrizieren. Die Beschriftung mit Titeln, Zahl- oder Prozentwerten sowie einer Legende erfolgt automatisch. Auch beliebiger zusätzlicher Text kann eingebaut werden. Eine Grafik, ist sie erst definiert, wird sich auch bei Zahlenänderungen selbständig mitverändern. Die Grafikfenster können sogar als Vektorgrafik (Metafile) abgespeichert werden, um sie z.B. in Calamus oder SciGraph weiterzuverwenden.

Zugegeben, die Auswahl an Grafikmöglichkeiten ist nicht gerade groß, und der Hinweis im Handbuch, daß man ja mit anderen Programmen weiterarbeiten könnte, zeigt, daß den Schöpfern von K-Spread nur ein Mindestmaß an Präsentationsgrafik vorschwebte. Aber einmal Hand aufs Herz, ist es wirklich nötig, zwischen unzähligen Variationsmöglichkeiten der Grafik wählen zu müssen? Schön wäre es zwar, aber in 80-90% aller Anwendungsfälle dürfte die derzeitige Ausstattung reichen. Vielmehr haben sich die Konstrukteure dieses Programms sinnvolle Arbeitshilfen einfallen lassen (z.B. völlig automatische Beschriftung der Grafikteile), die stärker zu Buche schlagen. Fazit zum Grafikteil: eher ein Plus.

Makros

Oft kann man sich dabei erwischen, daß die selben Arbeitsabläufe mehrmals vorkommen. Das wird natürlich dann ärgerlich, wenn es ganz stupide Wanderbewegungen im Arbeitsblatt sind; gehe zu, kopiere, markiere, gehe zu ... usw. Um diesem Frust abzuhelfen, gibt es eine recht nützliche Makrosprache, die es mit einer normalen Programmiersprache aufnehmen

könnte. Sinnvoller wird diese Einrichtung, wenn es z.B. darum geht, bestimmte Formeln regelmäßig in gleicher Art und Weise anzuwenden, ständig wiederkehrende Rechen-, Such- oder Vergleichsvorgänge zu starten, oder auch, wenn es nur darum geht, eine regelmäßige Datensicherung zu fahren. Im Extremfall ließe sich mit der Makrosprache eine eigene Benutzerführung basteln, so daß Datentypisten, die nichts anderes tun, als nur Daten einzugeben, vom Arbeitsblatt selbst nicht viel zu wissen brauchen, die richtigen Positionen steuert das Makro wie von Geisterhand selbständig an.

K-Spread kennt zwei Wege, um ein Makro zu definieren. Entweder man weiß, was man will, und schreibt die einzelnen Makrobefehle in ein speziell vorbereitetes Fenster von Hand, oder man überläßt dem Makrorekorder diese Arbeit. Gerade die Arbeit mit diesem Makrorekorder ist für Anfänger sehr angenehm: Aufzeichnung starten, alle Arbeiten ausführen, Aufzeichnung anhalten, fertig. Selbst für nachträgliche Änderungen ist der Makrotext jederzeit zugänglich, was bei umfangreichen (aufgezeichneten) Arbeitsvorgängen sehr nützlich ist. Übrigens: Makros können andere (und sich selbst - Achtung Endlosschleife - nützlich für Vorführungen) aufrufen, sogar ein Abarbeiten nach vorheriger Prüfung einer Bedingung ist machbar. Also alles fast wie eine große Programmiersprache, denn immerhin kennt die Makrosyntax fast 150 Befehle. Zusätzliches Bonbon: Auch die Grafik- und sogar GDOS-Funktionen sind per Makro steuerbar.

Beurteilung: Irgendwie müssen die K-Spread-Programmierer von den schlechten Beispielen der Branche durchaus Gutes gelernt haben. Bisher war es doch immer so, daß ein Makro im eigentlichen Arbeitsblatt definiert sein mußte (siehe LOTUS 1-2-3 u.ä.). In K-Spread wird ein eigenes Arbeitsblatt für Makros verwendet, das dann auch eigenständig abgespeichert und von anderen Rechenblättern aus verwendet werden kann. Urteil: sehr sinnvoll und der Befehlsvorrat kann sich sehen lassen.

GDOS in Verantwortung

Das hinlänglich (un)bekannte GDOS, eigentlich ein Grundbestandteil des Digital Research GEM, erhält in K-Spread 4 neue Ehren. Ursprünglich hätte GDOS ebenso wie die GEM-Benutzeroberfläche in den ROM-Bausteinen Ihres Atari Computers Einzug halten müssen. Weil aber das Rahmenbetriebssystem TOS (wer kennt es nicht?) viel zu umfangreich geworden ist (warum wissen die Götter), hat man großzügigerweise auf GDOS verzichtet. Programme die nun absolut nicht auf GDOS verzichten wollen (ironisch gemeint!), müssen dieses eben mit ausliefern und den Anwendern eine relativ umständliche Installationsprozedur aufzwingen. Glücklicherweise oder besser gesagt: nichtsdestotrotz haben die Erzeuger von K-Spread dem grafischen Gerätetreiber GDOS einige wichtige Aufgaben zugewiesen.

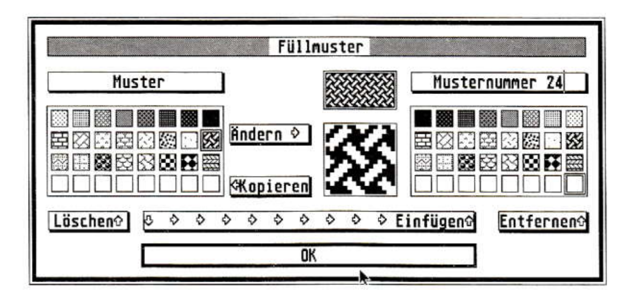


Bild 8: Nur ein Beispiel für die umfangreichen Dialogboxen

Um es aber gleich hier festzustellen, GDOS ist zum Betrieb von K-Spread nicht zwingend vorgeschrieben. Ist GDOS aber vor dem K-Spread-Start geladen worden (üblicherweise aus einem Auto-Ordner heraus), kann man alle GDOS-Fonts in die Arbeitsblattgestaltung mit einbeziehen. Sogar Proportional-Fonts und aufwendige Outline-Schriften werden korrekt behandelt.

Bis zum Jahresende wird die Firma Atari (hoffentlich) ihr neues FSM-GDOS der Allgemeinheit zugänglich gemacht haben. K-Spread hat seinen Test unter FSM-GDOS schon bestanden. Damit steht sogar der Verwendung von beliebig großen Vektor-Zeichensätzen nichts mehr im Wege.

Absolutes Bonbon unter GDOS: Jedem einzelnen Feld im Arbeitsblatt kann ein eigener GDOS-Font zugeordnet werden. Damit sind der freien Gestaltung Tür und Tor geöffnet. Überschriften erscheinen in wuchtig großen Lettern (wie es sich gehört), Randbemerkungen dürfen dagegen in winzigen Buchstaben ihre Information verbreiten, wichtige Ergebnisse könnten dann unterstrichen oder kursiv sein. Ein mitgeliefertes GDOS-Ausgabeprogramm bringt dann die ganze Kreation „WYSIWYG" (muß ich die doofe Abkürzung wirklich noch erklären?) auf das Druckerpapier. Das ist schon fast DTP (Desk-Tabellen-Publishing - kleiner Scherz des Autors).

Mein Urteil: Es ist nicht mehr als konsequent, daß die Programmierer nicht davor zurückschreckten, GDOS überhaupt zu verwenden. Vielen anderen Programmen wurde durch den GDOS-Mangel einiges an grafischer Vielfalt genommen, was die GEM-Oberfläche eigentlich von anderen Systemen abhob (nun gut, von Apples Mac und von Windows wollen wir erst gar nicht reden). K-Spread hat durch GDOS einiges an grafischer Reichhaltigkeit hinzugewonnen. Wertung: absolut positiv!

Jetzt kommt Farbe ins Spiel

Gerade in Richtung Farbenspiel hat der Produzentenkreis von K-Spread die Phantasie galoppieren lassen. Nicht nur, daß das Programm alle Groß- und Farbbildschrime unterstützt, Es läuft auch unter allen Auflösungen der Atari ST- und TT-Geräte. Aber in Richtung Farbenspiel hat man sich noch mehr ausgedacht. So kann man z.B. wahlweise negative Zahlen in der Tabelle automatisch rot darstellen lassen. Weiterhin sind die Schrift- und die Tabellenhintergrundfarbe für jede einzelne Zelle frei einstellbar. Nützlich würde so etwas sein, wenn man beispielsweise jede zweite Zeile mit einem anderen Farbhintergrund versähe, um eine große Tabelle mit kleiner Schrift übersichtlicher zu gestalten.

Insgesamt stehen vier verschiedene Farben mit jeweils vier Intensitätsabstufungen (Raster) zur Auswahl. Wertung: „Farbe ins Spiel" bringt dieses Programm wahrhaftig. Wenn die Farbwahl durch die Benutzer sinnvoll gehandhabt wird, kann sich dies als sehr nützlich für das Gesamtbild erweisen.

Auf gute Zusammenarbeit

„Sag', wie hältst es Du mit der Kompatibilität?" K-Spread kann nicht nur ASCII-Werte (und natürlich auch Texte) einlesen und automatisch den Zellen zuweisen, es hat sogar einen „Entwirr-Algorithmus" eingebaut. D.h. wenn eine Datei importiert werden soll, sieht K-Spread nach, ob irgendwelche Strukturen in der Datei auf eine gewissen Ordnung schließen lassen. Wird eine solche Ordnung (z.B. bestimmte Trennzeichen, absonderlich vieleLeer-, oder ASCII-Zeichen, die weder numerisch noch alphanumerisch sind) erkannt, so können diese sogenannten „zusammengeklebten" Datensätze automatisch wieder

auseinandergepflückt und auf einzelne Zellen verteilt werden.

Weiterhin versteht K-Spread das DIF-Format verschiedener Datenbankprogramme oder WKS/WRK von Lotus, LDW-Powerkalk sowie dergleichen aus den Programmen Lotus Symphony, DG-Calc und EasyBase, sowie verschiedener anderer Datenbanken und Omikron-BASIC. Diagramme übergibt K-Spread als Vektorgrafik an Calamus, als Metafile an SciGraph, und SciGraph macht daraus sogar eine PostScript-Datei.

Kleinigkeiten am Rande

Es sind (wie so oft) die Kleinigkeiten, die das Besondere eines Programms ausmachen. Bei K-Spread überwiegen etwas die „Plus-Kleinigkeiten":

1. Das Programm liegt in einer 285 kByte-Version gepackt vor, die sich auf 435 kByte „entblättert". Beide Versionen sind direkt startbar, wobei die gepackte (logischerweise) etwas länger braucht.
2. An die zwei Cursoren (stimmt diese Pluralform vielleicht?) muß man sich zwar erst gewöhnen, aber sonst sind sie sehr nützlich.
3. Auch das „Farbenspiel" dürfte der Übersichtlichkeit dienen und ist deshalb empfehlenswert, sofern man einen Farbmonitor betreibt.
4. Mit einem Paßwort kann der Zugriff auf einzelne Arbeitsblätter gesperrt werden.
5. Ein kleines Speicherinfofensterchen gibt Auskunft über den bisher verbrauchten und den noch freien Arbeitsspeicher. Diese Auskunft ist wichtig bei rasch anwachsenden Arbeitsblättern.
6. Zeileninhalte können in Spalten und umgekehrt transferiert werden. So könnte es z.B. vorkommen, daß man eine horizontal angeordnete Zahlenkolonne aus der Datenbank vertikal in der Tabelle haben möchte.
7. Die Weitersprungfunktion ist in alle vier Richtungen einstellbar. D.h. wenn nach einer Eingabe die Return-Taste gedrückt wird, würde das Programm lediglich die Zelle schließen, und der Cursor bliebe stehen. Bei der Weitersprungfunktion geht der Cursor nach der Return-Taste in eine Nachbarzelle weiter. Sehr nützlich beim Eingeben langer Zahlenreihen.

Kommen wir aber auch einmal zu den „Minus-Kleinigkeiten":

1. Es gibt keine Hilfe-Funktion. Vielleicht, so dachten sich die Programmierer, ist sie auch gar nicht nötig, quasi als Rechtfertigung für das außergewöhnlich beleibte Handbuch? (Übrigens: In der großen weiten MS-DOS-Welt sind die sogen. Online-Hilfen ohnehin wieder out.)
2. Beim Ausstieg aus dem Programm wird die Tabelle „gecleart". D.h. es dauert eine geraume Zeit (jenachdem, wie voll die Tabelle war), bis K-Spread alles weggeräumt hat. Warum das so ist, konnte ich nicht ergründen.
3. K-Spread unterstützt nicht den arithmetischen Coprozessor 68882 im Atari TT, und damit steht es in trautem Einklang mit allen anderen Tabellenprogrammen für den ST/TT.
4. Das oben (5.) erwähnte Speicherinfofensterchen sollte unbedingt als permanent in der Arbeitsoberfläche sichtbar sein.

Epilog

K-Spread ist fast schon ein integriertes Paket. Ausgehend von der Tabellenkalkulation, stellt es ausreichend viele Funktionen für eine Datenbank und als Textverarbeitung zur Verfügung. Der Grafikteil ist im Umfang durchaus als befriedigend einzustufen. Ein sehr ausführliches Handbuch erklärt auf über 300 Seiten alles, was wichtig ist. Für einen ausreichenden Bilderdurchsatz, der die Theorie angenehm auflockert, ist gesorgt.

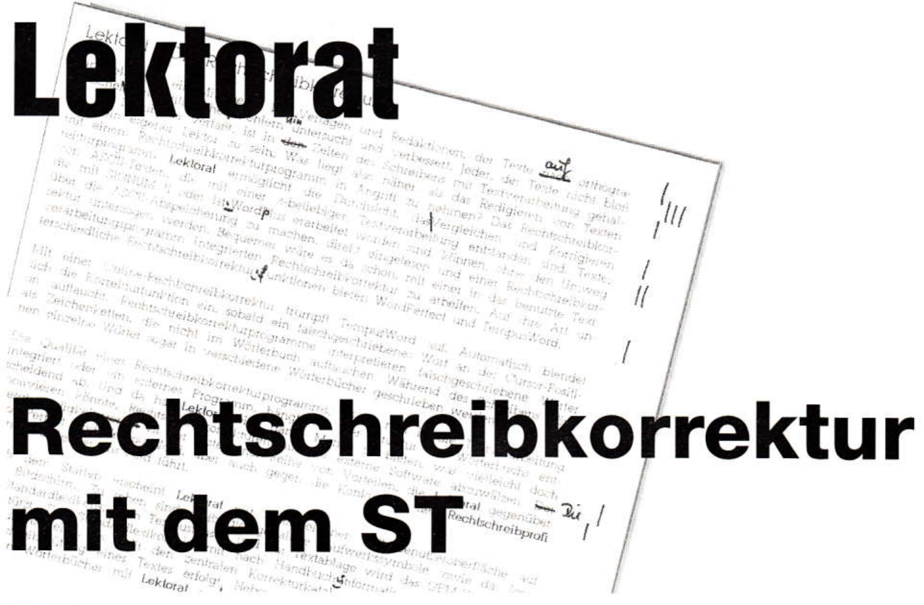
Ich scheue mich davor, so subjektive Aussagen wie „hohe Rechenzeit", „schnelles Scrolling", „guter Grafikteil" oder „sauber programmiert" in eine abschließende Bewertung mit einfließen zu lassen. Da müßte ein Vergleichstest zeigen, wie relativ diese Metaphern wirklich sind. Für den Anwender ist einzig und allein interessant, ob ihm mit K-Spread 4 ein nützliches Werkzeug an die Hand gegeben wird, und ob es ihm bei seiner Problemlösung behilflich sein kann.

Die kleinen Schwächen, die K-Spread 4 in seiner englischen Version noch hatte, sind nun völlig ausgemerzt, und es darf mit Fug und Recht behaupten eine runde Sache zu sein. Die, noch verbliebenen kleinen Schönheitsfehler (wer hat sie nicht?), tun dem durchweg positiven Gesamtbild keinen Abbruch und für DM 248 stellt sich hier ein Herausforderer, mit dem zu rechnen ist (im zweideutigen Sinne des Wortes).

DK

Bezugsquelle:
Omikron Soft- und Hardware GmbH
Sponheimstraße 12
W-7530 Pforzheim
Tel: (07231) 35 60 33

Lektorat



Rechtschreibkorrektur mit dem ST

Ein Lektor ist ein Mitarbeiter bei Verlagen und Redaktionen, der Texte auf orthographische und stilistische Fehler hin untersucht und verbessert. Jeder, der Texte nicht bloß für die Schublade verfaßt, ist in Zeiten des Schreibens mit Textverarbeitung gehalten, sein eigener Lektor zu sein. Was liegt also näher, als das Redigieren von Texten mit einem Rechtschreibkorrekturprogramm in Angriff zu nehmen? Das Rechtschreibkorrekturprogramm Lektorat ermöglicht die Durchsicht, das Vergleichen und Korrigieren von ASCII-Texten, die mit einer x-beliebigen Textverarbeitung entstanden sind. Texte, die mit SIGNUM2 oder Wordplus erarbeitet worden sind, können, ohne den Umweg über die ASCII-Abspeicherung zu machen, direkt eingelesen und einer Rechtschreibkorrektur unterzogen werden. Bequemer wäre es da schon, mit einer in das benutzte Textverarbeitungsprogramm integrierten Rechtschreibkorrektur zu arbeiten. Auf ihre Art unterschiedliche Rechtschreibkorrekturfunktionen bieten WordPerfect und TempusWord.

Mit einer Online-Rechtschreibkorrektur trumpft TempusWord auf. Automatisch blendet sich die Korrekturfunktion ein, sobald ein falschgeschriebenes Wort an der Cursor-Position auftaucht. Rechtschreibkorrekturprogramme interpretieren falschgeschriebene Wörter als Zeichenketten, die nicht im Wörterbuch auftauchen. Während des Schreibens können einzelne Wörter sogar in verschiedene Wörterbücher geschrieben werden.

Die Qualität eines Rechtschreibkorrekturprogramms, sei es nun in die Textverarbeitung integriert oder ein externes Programm, hängt von der Struktur des Wörterbuchs entscheidend ab. Und da hat Lektorat eine ganze Menge zu bieten, was vielleicht doch motivieren könnte, Rechtschreibkorrekturen auf die externe Software abzuwälzen. Die Geschwindigkeit ist der erste einer Reihe von Vorteilen, die Lektorat gegenüber WordPerfect und TempusWord, aber auch gegen die Konkurrenz, den Rechtschreibprofi von Data Becker, ins Feld führt.

Nach dem Starten erscheint Lektorat mit einer eigenen GEM-Benutzeroberfläche auf dem Bildschirm. Zu sehen sind großzügig gestaltete Laufwerkssymbole sowie das Icon des Standardlexikons. Zum Textaustausch und zur Textablage wird das GEM-Klemmbrett unterstützt. Das Standardlexikon ist mit nach Handbuchinformation 110.000 Einträgen sehr umfangreich. Es stellt den zentralen Korrekturkatalog dar, auf dessen Basis die Rechtschreibprüfung eines Textes erfolgt. Neben dem Standardlexikon können bis zu vierzehn Wörterbücher mit Lektorat neu angelegt und mit Wortneuaufnahmen erweitert werden.

Neu angelegte und angemeldete Wörterbücher werden als Dateisymbole auf der Lektorat-Benutzeroberfläche dargestellt. Zur Rechtschreibprüfung wird das Icon des zu prüfenden Textes mit der Maus auf das Standardlexikon-Icon geschoben. Sofort beginnt die Phase der Rechtschreibkorrektur mit dem alphabetisch sortierten Einlesen der Textwörter, beginnt die Rechtschreibprüfung beim ersten Wort der Textdatei und vergleicht nacheinander alle Wörter des Textes mit dem Einträgen des Wörterbuchs. Im Prinzip handelt es sich bei dieser Technik um ein Vergleichen von Zeichenketten mit alphabetischer Struktur. Jeder Buchstabe eines Wortes wird Zeichen für Zeichen mit den systematisch aufgelisteten Einträgen des Wörterbuchs verglichen, bis die Identität der Zeichen oder eine Abweichung festgestellt wird. Bei Feststellung einer Abweichung meldet sich auf der Stelleder Korrekturmodus und mahnt die Beseitigung des Fehlers an. Mit der Einstellung verschiedener Korrekturparameter könnte der Korrekturvorgang noch beschleunigt werden. Muß zwischen Groß- und Kleinschreibung unterschieden werden? Wie sind Wortsonderformen und Abkürzungen zu bewerten? Ist es sinnvoll, die Silbentrennung zu überprüfen, wenn der ASCII-Text doch mit einem anderen Textverarbeitungsprogramm formatiert und druckfertig vorbereitet wird? Sind Sonderformen und Abkürzungen zu prüfen, und benötigtLektorat ein zweites Wörterbuch, das Sonderfälle deutscher Sprache verzeichnet hat? Im Prinzip wäre es auch möglich, fremdsprachige Sonderwörterbücher anzulegen und zweisprachige Textdateien prüfen zu lassen.

Ein normaler Arbeitsgang mit Rechtschreibkorrektur einer Textdatei sieht folgendermaßen aus. Zunächst ist eine ASCII-Textdatei zu laden. Zur Behandlung von SIGNUM!-Dokumenten können die Zeichencodes von SIGNUM-Zeichensätzen in Konversionstabellen definiert werden. Dann könnte die Rechtschreibprüfung auch von SIGNUM!-Dokumenten losgehen.

Wenn im Korrekturtext alles in Ordnung ist und Sie Ihr guter persönlicher Lektor gewesen sind, meldet sich das Korrekturmenü nicht. Mit der Maus führt man das Text-Icon auf das Symbol des Standardlexikons. Der Korrekturlauf beginnt und zeigt lediglich mit einem knappen Dialog die aktuell bearbeitete Buchstabengruppe an. Das Vergleichen des Wortmaterials mit den Wörterbucheinträgen geschieht sehr schnell, so daß man sich nur wundern kann, was für eine Nebensächlichkeit die Rechtschreibprüfung darstellt. Wenn Wörter aber nicht wohl gewogen und orthographisch korrekt geschrieben sind, tritt das Lektorat-Korrekturmenü auf den Plan. Jedes fehlerhaft geschriebenes Wort wird mit lexographischen Einträgen mindestens im Standardlexikon verglichen. Entweder stellt Lektorat die Identität von Textwort und Eintrag fest, dann wird der Korrekturlauf kommentarlos fortgesetzt; oder die Schreibweise des Textwortes wird nicht als Lexikoneintag verifiziert. Mit einem Korrekturmenü stellt Lektorat dessen Einzigartigkeit fest, was auf einen Rechtschreibfehler schließen läßt, sofern es sich um keinen Eigennamen handelt.

Im Bild haben Sie ein Beispiel für einen Korrekturfall. Über Tastatur können Sie den Rechtschreibfehler korrigieren und mit

der Einfüge-Option in den Text schreiben. Wichtig ist auch die Option „Merken", mit welcher Wörter in einer Zwischenablage gespeichert werden. Nach dieser Methode gesammelte Wörter können später auf verschiedene Lektorat-Wörterbücher verteilt werden. Diese Funktion ist u.a. für alle Lektorat-Benutzer interessant, die Spezialwörterbücher mit Fachterminologien anlegen wollen. Bei der Rechtschreibprüfung können mehrere Wörterbücher parallel aufgeschlagen sein. Lektorat durchsucht im Prüffall nacheinander die aktiven Wörterbücher. Wenn kein Eintrag gefunden wurde, erscheint ein entsprechender Hinweis, bzw. bei Richtigkeit wird der Prüfvorgang kommentarlos fortgesetzt. Den Gesamtwortschatz kann man also durch Anlegen verschiedener Wörterbücher zum einen differenzieren und zum anderen erheblich erweitern.

Leider verfügt das Korrektur-Menü über keine Doppelwort-Prüfung, so daß hier die Aufmerksamkeit des Korrekturleser besonders gefragt sein wird. Wünschenswert wäre zudem ein Korrekturmodus, der es ermöglichte, daß bestimmte Wörter bis zum Textende keiner wiederholten Prüfung unterzogen würden. So macht sich beim Korrekturlauf nachteilig bemerkbar, daß die Prüfung der Groß-/Kleinschreibung der Anredefloskel >Sie< wiederholt wurde. Es ist nicht nötig, solche Wörter, wiederholte Begriffe mit besonderen Schreibweisen oder Namen dauernd zu prüfen. Das führt nur zu zeitlichem Aufenthalt.

Die Korrekturleistung eines Rechtschreibkorrekturprogramms liegt sicherlich in seinem methodischen Aufbau und dem Prüfverhalten bei grammatikalischen Beugungsformen. Die Erkennung von Deklinations- und Konjugationsformen ist ein wichtiges Kriterium für ein sinnvolles Textlektorat. Relativ problemlos wäre es, ein einfaches Substantiv wie >Verband< in einem Text mit dem entsprechenden Wörterbucheintrag zu vergleichen. Was ist aber mit einigen assoziativ sich einprägenden Ableitungsformen: Verbände - verbinden - verbinde - verbindet - Verbinder - verbunden - Verbund - Verbundene - Verbünde? Diese Liste könnte noch verlängert werden und verdeutlicht die Schwierigkeiten, vor die eine Rechtschreibprüfung gestellt ist. Es geht nicht nur um die Feststellung der Abweichung von der offiziellen Rechtschreibnorm, sondern auch um die Erfassung der grammatischen Feinstruktur eines Wortes. Daß hier wirklich kreative Spracharbeit geleistet worden ist, macht Lektorat mit seinen gutstrukturierten Verzeichnissen der Deklinations- und Konjugationsformen sichtbar.

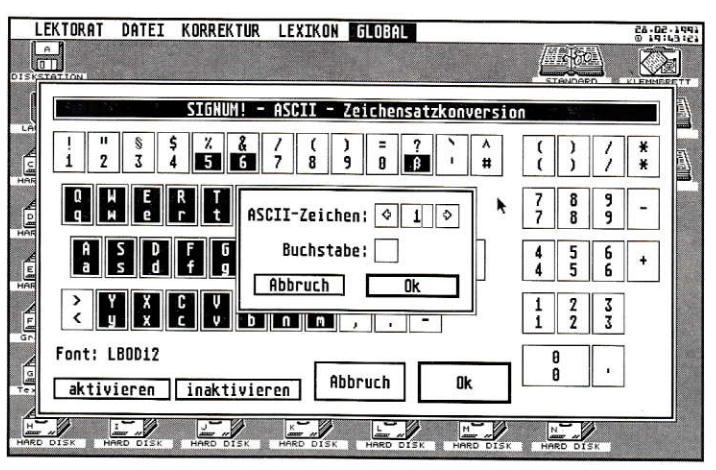


Abb. 1: Vor der Rechtschreibkorrektur von Signum!-Dokumenten steht die Konversion der dokumentinternen Zeichensätze.

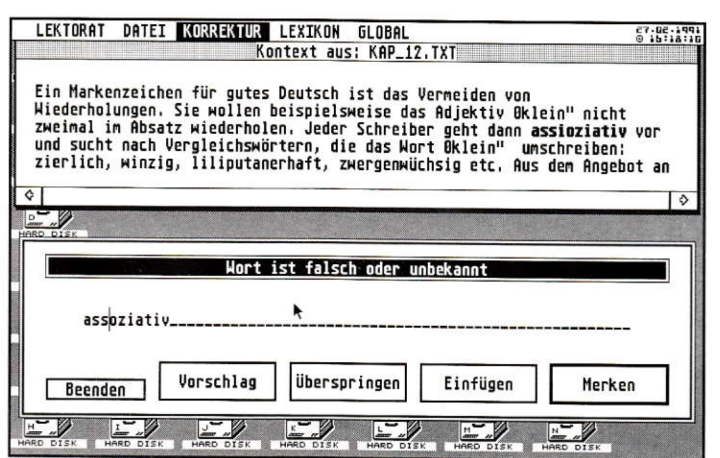


Abb. 2: Lektorat hat einen Rechtschreibfehler entdeckt!

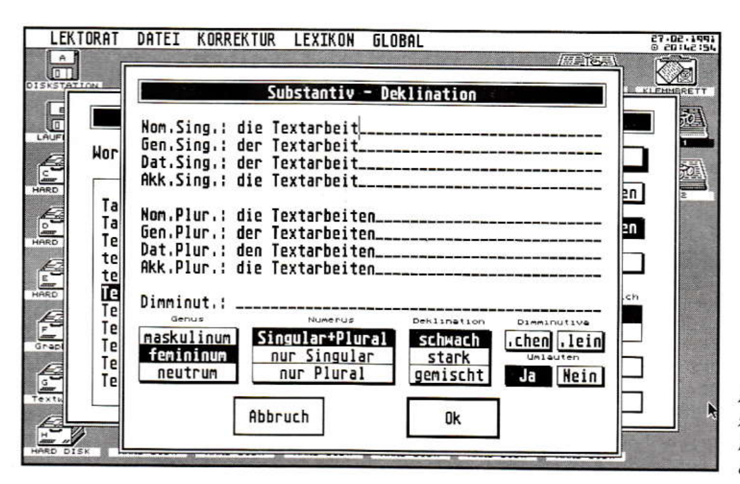


Abb. 3: Das Menü zum Bestimmen der Deklinationsformen eines Substantivs

Ihre Grammatik können Sie im Bücherschrank lassen. Die Deklinationsformen eines Substantivs/Adjektivs bzw. Konjugationsformen eines Verbs beherrscht Lektorat souverän.

Die Verwaltung der Wortformen übernimmt der Menüpunkt „Lexikon". Hier wird am deutlichsten, daß Lektorat ein echtes Werkzeug für die intensive Rechtschreibprüfung ist. Zunächst öffnet man ein Lexikon, das bearbeitet werden soll. Das könnte entweder das Standardlexikon oder ein Spezialwörterbuch sein. Spezialwörterbücher könnten beispielsweise Fachbegriffe oder Vokabeln verzeichnen, die man im Standardlexikon vergeblich sucht. „Strukturierte Lexikonverwaltung" heißt, daß man dem angebotenen Inventar der Wörterbücher nicht hilflos ausgeliefert ist, sondern selber Wörter ändern oder ergänzen darf. Wortsammlungen zu speziellen Wortfeldern anzulegen und zu untersuchen, ist ein sehr vornehmes Arbeitsziel, für das Lektorat, unabhängig von der Rechtschreibprüfung, wie prädestiniert scheint. Aber das wäre nur eine von vielen Möglichkeiten, die der Menüpunkt „Lexikon" zu bearbeiten erlaubt.

Eine einfache Such-Option ermöglicht die Wortsuche im aufgeschlagenen Lexikon/Wörterbuch. Wenn man die Hinzufüge-Option aktiviert, wird das Wort in die alphabetische Struktur des Wörterbuches eingegliedert und bei späteren Recht-

schreibprüfungen berücksichtigt. Ein Schatzkästlein für Sprachversessene und solche, die es werden wollen, ist der Menüpunkt „Wortformen", auf den bereits hingewiesen worden ist. Er erlaubt das Bearbeiten von Deklinations- und Konjugationstabellen. Wer seine Grammatikkenntnisse aufrischen möchte, kann sich in vielen Stunden mit deutschen Ablautreihen, schwachen und starken Verben, Zeitverhältnissen und Zeitformen oder Wortsteigerungen von Adjektiven beschäftigen. Diese Tabellen sind Vorschlagsangebote, die bei der Rechtschreibprüfung berücksichtigt werden.

Das Handbuch bietet einige Anschauungsbeispiele, die erklären, wie Lektorat Wortformen verwaltet. Das scheint aber bei der Komplexität und dem Wortreichtum der deutschen Sprache nur ein Tropfen auf den heißen Stein zu sein. Man könnte diesen Programmteil, der sich mit grammatikalischen Fragen beschäftigt, auch zum Lernen der deutschen Konjugations- und Deklinationsreihen zweckentfremden. Der Kaufanreiz für Lektorat wäre nur noch größer.

Auf maximal fünfzehn Wörterbüchern können unbekannte Wörter mit Bezeichnung der entsprechenden Wortart eingetragen werden. Bedingung ist, daß das Wörterbuch erzeugt und mit dem Zugriffspfad angemeldet worden ist.

Lektorat ist ein kleines Programm, das auf einer Diskette mit einer knappen Dokumentation ausgeliefert wird. Besonders über die Methode der Verwaltung von Wortlisten fehlen mir einige Detailinformationen, was aber nicht daran hindert, mit der Rechtschreibprüfung von Texten zu beginnen. Weil das Schreiben auf Basis grafischer Zeichensätze mit ST-Textverarbeitungen sehr beliebt ist, ist eigentlich schade, daß bloß SIGNUM-Dokumente einer Rechtschreibprüfung unterzogen werden können. Wer denkt an Tempus-Word, Script oder That's Write?

Eindeutiger Trumpf von Lektorat ist die schnelle und sichere Rechtschreibprüfung mit Korrekturmodus. Der zeitliche Nachteil muß dafür aber in Kauf genommen werden. Man importiert eine ASCII-Textdatei zu Prüfzwecken nach Lektorat, um sie dann nach den Regeln der grammatischen Künste (Syntaktik, Semantik, Pragmatik) zu bearbeiten, und erhält retour einen fehlerbereinigten und geprüften Text. Diesen braucht man nur noch als ASCII-Datei zu speichern und in das ursprüngliche Textformat zu konvertieren.

Ralf Blittkowsky

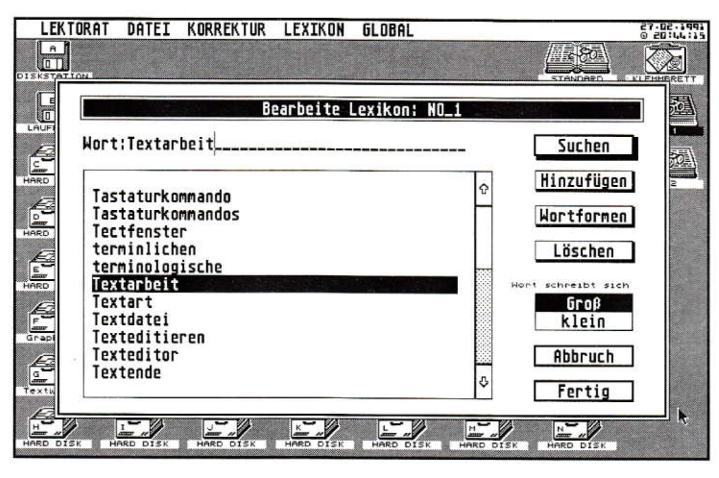


Abb. 4: Viele Möglichkeiten, die über die reine Rechtschreibprüfung hinausgehen

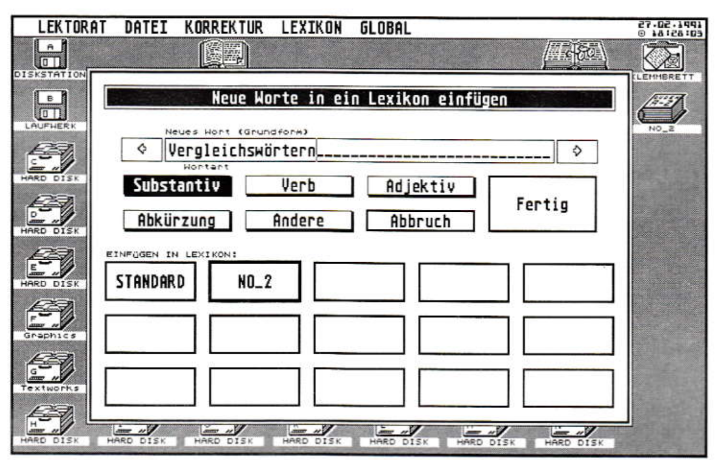
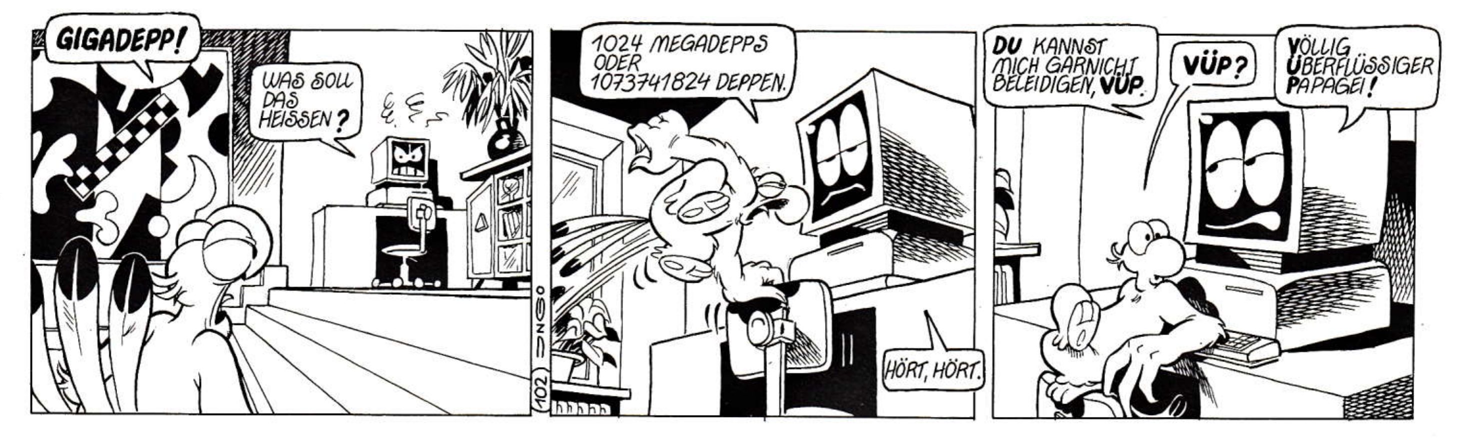
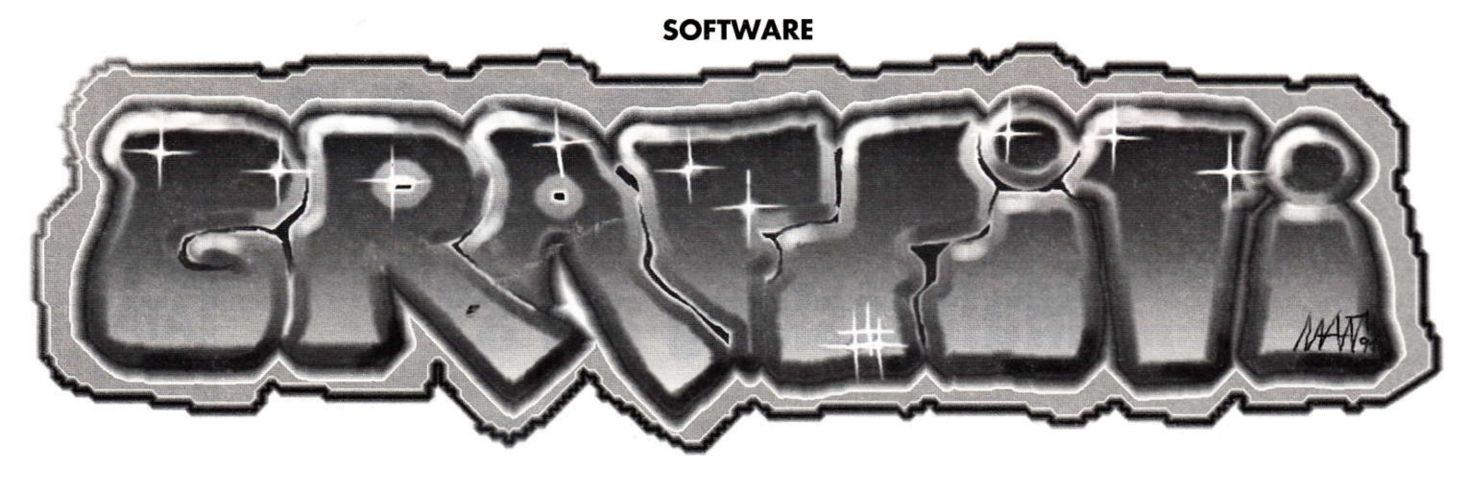


Abb. 5: In welches Wörterbuch soll das Wort?

Bezugsadresse:
Lektorat GbR Kubasch & Stahl
Dachsbergstr. 6b
W-3500 Kassel
Tel. (0561)37463







Betonsprüher zwischen zwei Welten

Graffiti - da denkt man unwillkürlich an die Sprühdosenkünste, die die Berliner Mauer schmückten, als sie noch nicht abgerissen war. Schnell wachsen hier Assoziationen an bunte, mitunter wirre Malaktionen. Die Software Graffiti der K&L-Datentechnik GbR dagegen ist keineswegs bunt, sondern einzig und allein für die monochrome Bildschirmauflösung gedacht.

Hier bieten sich dem Käufer zweierlei Programmfunktionen an - Shell und Zeichenprogramm. Welche Vorteile oder Neuerungen in diesem Programm stecken, sollte sich in einem Test zeigen.

Was ist eine Shell?

Beim Wort „Shell" mag mancher denken: „Was wollen die denn schon wieder, ich tanke seit Jahren bei Aral..." Mit Tankstellen hat der Begriff jedoch ebensowenig zu tun wie mit Maria Schell. Eine Shell (englisch: Muschel) ist eine Art Benutzeroberfläche, die mehrere Programme verwaltet. Durch einfachen Knopfdruck werden beliebige Anwendungen ohne lästiges Öffnen von Fenstern und Pfaden gestartet.

Graffiti ist eine grafikorientierte Shell und stellt die einzelnen Programme in Form von Symbolen dar, die auch Icons genannt werden. Diese muß der Benutzer zuvor einrichten und kann dann bis zu 24 Programme per Mausklick aufrufen. Zudem verwaltet die Software zwischen einer und maximal neun Grafikseiten, die bis zu DIN A4 groß sein dürfen. Besser erscheint uns eine Lösung, weniger Seiten zu ermöglichen, dafür aber die Maximalgröße zumindest auf DIN A3 zu erweitern. Man wird sehen, was neue Programmversionen in dieser Richtung bringen. Auf Wunsch können die Seiten verkleinert nebeneinander angezeigt werden. Die höchstmögliche Auflösung beträgt 400 dpi (englisch: dots per inch = Punkte pro Zoll). Da die Grafikseite immer im Speicher gehalten wird, ist ein Mindestspeicherbedarf von zwei Megabyte Voraussetzung. Dies ist durch die ganzseitigen Grafiken bedingt und somit unvermeidbar. Bei mehreren Seiten mit hoher Auflösung steigt der Speicherbedarf schnell an. So wären bei neun Seiten mit einer Auflösung von 400 Punkten pro Zoll theoretisch etwa 17 Megabyte notwendig...

Optisch lehnt sich unsere „Sprühdosenkunst"-Software stark an das Programm Calamus an. Die Idee, mehrere kleine Symbole am linken Bildschirmrand zur Funktionssteuerung zu nutzen, wurde in ähnlicher Funktionsweise gestaltet. Im Zeichenprogramm wird dies noch deutlicher. In der „Muschel" werden sämtliche zu startenden Programme plaziert, die Graffiti verwalten soll. Leider funktioniert das Nachladen von Fremdprogrammen nicht immer fehlerfrei. Gerade dann, wenn bestimmte Dateien aus dem aktuellen Pfad nachgeladen werden müssen (Druckertreiber, Resource-Dateien), kann es zu Komplikationen kommen.

Zusätzlich zu den Symbolen am linken Rand finden sich am oberen Bildrand noch einige Extrafunktionen (siehe Bild 1). So lassen sich hier etwa Mausparameter einstellen oder Grafikseiten anzeigen/löschen oder ausdrucken. Ferner erscheint am oberen Bildrand in der Shell eine kleine, analoge Uhr, die die aktuelle Systemzeit anzeigt - ein recht nützliches Utility. Im Lieferumfang sind weiterhin verschiedene Module (eine Art Zusatzprogramm) enthalten, die bestimmte Arbeiten verrichten:

- Paint ist ein Zeichenprogramm
- Drehen kann IMG-Grafiken drehen
- PF-Edit ist ein Zeichensatzeditor
- Icon-Edit erstellt die Symbole
- Hardcopy druckt eine Seite aus
- Install stellt Optionen ein

Die Modultechnik bringt einige Vorzüge mit sich. Auf der einen Seite ist das Programm für neue Module und Erweiterungen offen. Zudem wird der Speicherbedarf in mehrere Einzelstücke reduziert. Auf der anderen Seite hat der Kunde die Wahl, nur die Module zu kaufen, die er tatsächlich benötigt. Er ist nicht gezwungen, für ein „Komplettpaket" mit selten oder nie benutzten Programmteilen mehr Geld auszugeben als nötig.

Zur Zeit scheint es eine Modewelle zu sein, den Systemzeichensatz des Computers durch eigene Zeichensätze zu ersetzen. Wer's mag - wieso eigentlich nicht. Dennoch ist es mitunter äußerst störend, wenn der Zeichensatz beim Aufrufen anderer Programme nicht abgeschaltet wird. Gerade bei einer Shell werden viele „Fremdprogramme" aufgerufen. Da ist es doch verwirrend, wenn in der Zeichensatztabelle von Wordplus plötzlich der Zeichensatz der Shell verwendet wird.

Das Programm funktioniert bisher nur in der monochromen Auflösung von 640x400 Punkten. Angeblich ist eine An-

passung an Atari TT und Großbildschirme aber bereits geplant.

Zukunftsperspektiven

Insgesamt ist die Shell mit Icons im Stil von Calamus ansprechend gelöst, hat vermutlich aber wenig Zukunft. Um als vielverwendete Shell eingesetzt werden zu können, müßten sich andere Software-Hersteller an Graffiti anpassen. Dies ist jedoch kaum wahrscheinlich, da die Shell in ihrem Aufbau zu unflexibel ist. Ihr Einsatz ist an und für sich nur im grafischen Bereich zu suchen. Insofern werden wohl andere Firmen kaum auf eine Graffiti-Programmierung umsteigen. Dagegen spricht auch die bisher noch sehr geringe Verbreitung von Graffiti. Man kann also davon ausgehen, daß nur die Firma K&L-Datentechnik das Graffiti-Konzept mit Modulen weiterhin unterstützen, erweitern und aufrechterhalten wird.

Handbuch

Das Handbuch geht auf 73 Seiten sehr ausführlich auf fast alle Programmpunkte ein. Mitunter stimmen die Angaben des Handbuches jedoch nicht ganz mit der tatsächlichen Programmbedienung überein. So heißt es etwa, daß zum Zeichnen einer Linie die Maustaste zum Festlegen des Endpunktes gedrückt gehalten werden muß, in Wirklichkeit sind jedoch zwei Mausklicks (einer für Anfang und einer für Ende) notwendig. Meist handelt es sich jedoch um Kleinigkeiten, die beim Umgang mit dem Programm schnell klar werden.

Um dem Benutzer den Einstieg in Graffiti zu erleichtern, wurde das Handbuch zahlreich bebildert. Wohl, um es nicht zu lang werden zu lassen, sind die Abbildungen oft so klein geraten, daß sie schlecht zu erkennen sind. Dies wird durch ein ausführliches Sachwortverzeichnis am Ende des Handbuches jedoch wieder wettgemacht. So fällt auch das Nachschlagen einzelner Funktionen nicht schwer.

Paint

Das Modul Paint ist ein vollständiges Zeichenprogramm. Daß Graffiti aus den Händen von Konstantinos Lavassas und Thomas Klingelhöfer stammt, läßt erwarten, daß sich einige Programmfunktionen aus ihrem Vorgängerprodukt „Lavadraw" wiederfinden. Hierauf muß der Käufer nicht lange warten - schon beim ersten „Reinschnuppern" in die Untermenüs fallen zahlreiche Möglichkeiten auf, die Lavadraw mehr als ähneln.

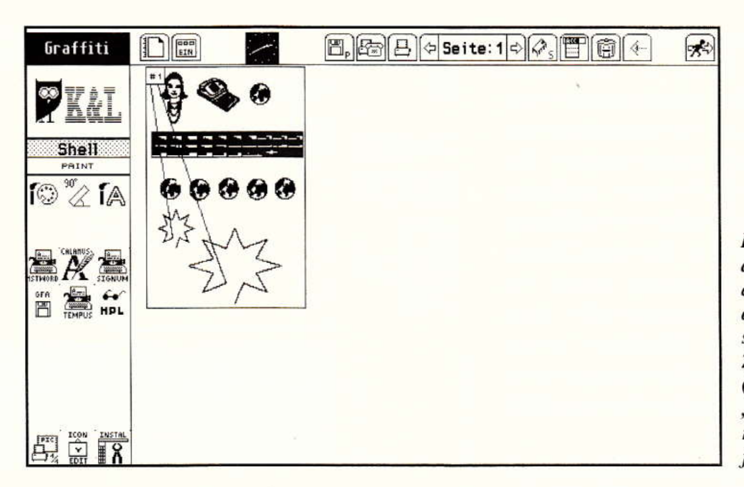


Bild 1: Graffiti dient als Shell und kann andere Programme auf Knopfdruck starten. Manche Zeichenfunktionen (zum Beispiel „Stern") sind jedoch noch nicht ganz fehlerfrei.

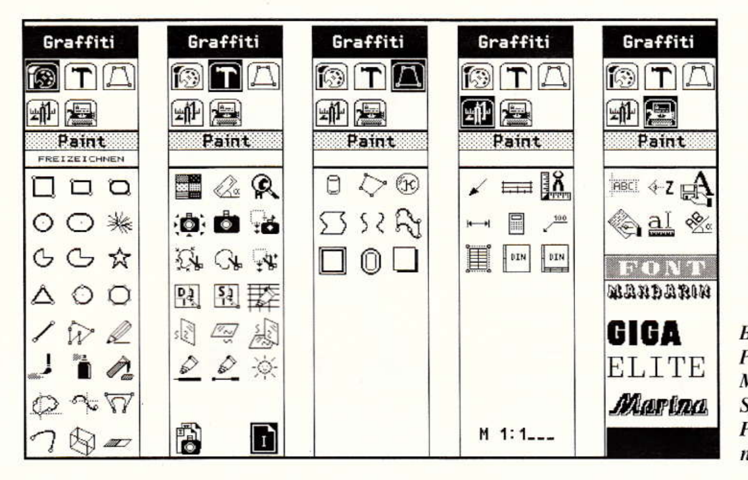


Bild 2: Im Modul Paint wurde das Menüsystem vieler Symbole dem DTP-Programm Calamus nachempfunden.

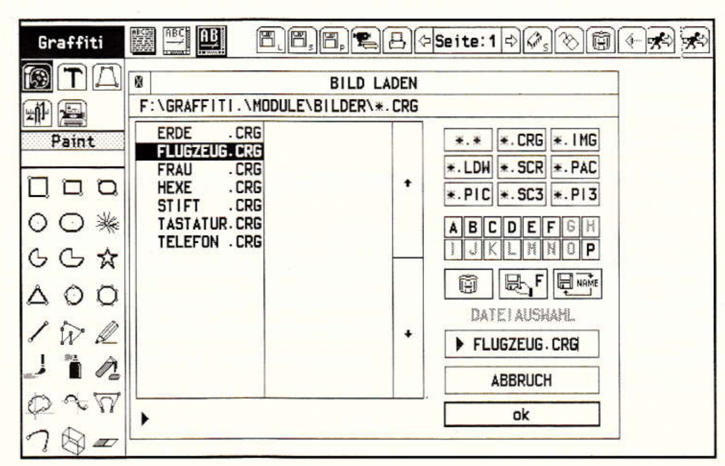


Bild 3: Die neue Dateiauswahlbox (Fileselector) gibt die Dateieindungen einladbarer Grafikstandards vor und kann bis zu 30 Einträge anzeigen.

Ganz anders dagegen ist die Menüsteuerung ausgefallen. Wie oben erwähnt, haben sich die Programmierer am Menüsystem der DTP-Software Calamus orientiert. In gleicher Art und Weise finden sich hier in verschiedenen Hierarchie-Ebenen Haupt- und Untermenüs wieder (siehe Bild 2). Dieses „etwas andere" Menü ist im Gegensatz zu „herkömmlichen" Menüleisten anfangs zwar ein wenig gewöhnungbedürftig, nach kurzem Arbeiten erweist es sich jedoch als sehr effektiv. Eine Referenzkarte, auf der die fünf Hauptmenüs mit ihren jeweiligen Funktionen abgebildet sind, wäre eine sehr nützliche Hilfe. Die Position der Menüeinträge ließe sich so noch leichter einprägen. Laut Handbuch wurde versucht, in allen Modulen gleiche Symbole für gleiche Funktionen zu verwenden. Leider trifft dies mitunter nicht auf die Einheitlichkeit in der Bedienung zu. Ungereimtheiten gibt es zum Beispiel, weil das Verlassen des Programmes mal mit der rechten Maustaste bestätigt werden muß, dann aber wieder mit einem Klick in eine Dialogbox.

Neben den üblichen Standardfunktionen bietet Graffiti mehrere Kopiermöglichkeiten und Effekte. Weiterhin stehen mehrere Texteingabearten für ein- oder mehrzeiligen sowie gedrehten Text zur Verfügung. Im Speicher lassen sich bis zu fünf Zeichensätze halten. Im Editor können per Tastenkombination alle ASCII-Zeichen eingegeben werden. Folglich auch solche, die sonst auf der Tastatur nicht

erreichbar sind (zum Beispiel das Copyright-Zeichen Nr. 189). Ist der Text jedoch erst einmal in die Grafikseite geschrieben, kann der aktuelle Zeichensatz problemlos durch einen neuen ersetzt werden. Dadurch lassen sich auf einer Grafikseite beliebig viele Zeichensätze verwirklichen. Beim Freihandzeichnen fällt dem Benutzer auf, daß das Programm bereits bei mittelschnellen Bewegungen der Maus nicht mehr mitkommt. Hier kann nur bei extrem langsamen Mausbewegungen eine starke Treppenbildung der Freihandlinie vermieden werden.

Als besonders angenehm fällt die vielseitige Kompatibilität zu anderen Grafikprogrammen auf. Neben weit verbreiteten Bildformaten der Programme Degas (*.PI3), STAD (*.PAC) und dem Grafikstandard mit 32000 Bytes (*.PIC / *.SCR) lassen sich auch GEM-Images (*.IMG), Calamus-Rastergrafiken (*.CRG) und Lavadraw-Seiten (*.LDW) laden und vor allem auch speichern. Somit dürfte die Anbindung an andere Programme optimal verwirklicht worden und kein Wunsch offen geblieben sein. Zum Laden wurde eine eigene Dateiauswahlbox (Fileselector) integriert, die die verschiedenen Bildformate auf Knopfdruck anzeigt (siehe Bild 3). Zudem lassen sich bis zu 30 Einträge gleichzeitig in der Box darstellen. Das Bildformat kann auch erkannt werden, wenn die Endung des Dateinamens nicht korrekt gesetzt wurde. Wird ein Bild eingeladen, wird zuerst die Größe ermittelt und dann ein Rahmen auf der aktuellen Grafikseite angezeigt. So ist ein Positionieren der Grafik an beliebiger Stelle möglich. Sollte beim Abspeichern der freie Diskettenplatz ausgehen, kann der Benutzer mit Graffiti-Paint auch eine Diskette formatieren, ohne das Programm verlassen zu müssen. Jeder Nutzer wird dies begrüßen.

Die meisten Optionen, etwa für Liniendicke, Füllmuster und ähnliches, lassen sich nur per Tastendruck aufrufen. Dafür kann aber während des Zeichnens noch schnell das Füllmuster verändert werden. Auch dies erweist sich oftmals als vorteilhaft. Als sehr positiv sind ferner die vielfältigen Optionen hervorzuheben, von denen nur einige kurz aufgelistet werden sollen. So kann etwa eingestellt werden, ob ein fertig erstelltes Objekt sofort nach dem Zeichnen in die Seite übernommen werden oder zuvor noch bewegt werden soll. Die Wahl der Kopierrichtung zwischen beliebig, horizontal oder vertikal ist gerade beim Erstellen von grafischen Tabellen sehr hilfreich. Bei Texten können der Zeichen- ebenso wie der Zeilenabstand und das Spacing (automatisches Anpassen der Zwischenräume) verändert

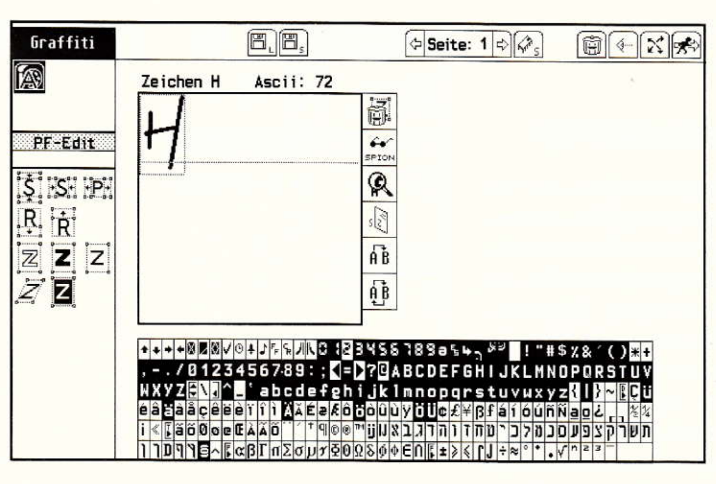


Bild 4: Der Zeichensatzeditor bietet einige Funktionen, um Buchstaben und Ziffern in ihrem Aussehen zu verändern. Nötige Grundfunktionen wie Linie, Rechteck oder Kreis wurden dabei ganz vergessen.

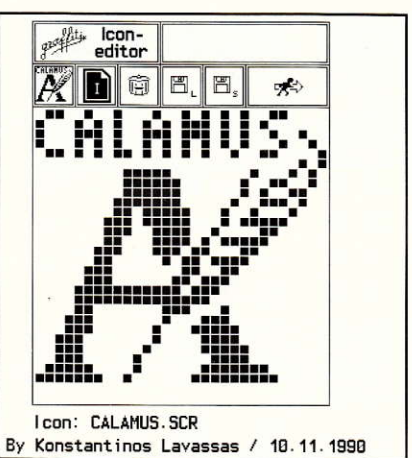


Bild 5: Der Icon-Editor soll beim Erstellen von eigenen Symbolen behilflich sein. Er ist jedoch so spartanisch ausgestattet, daß sich nur Punkte setzen oder löschen lassen.

werden. 72 verschiedene Füllmuster finden gleichzeitig im Programm Platz. Werden mehr benötigt, kann ein neuer 72er-Satz nachgeladen werden.

Darstellungsarten

Den größten Vorteil gegenüber anderen Grafikprogrammen bietet Graffiti in der komfortablen Bearbeitung ganzer Seiten. Da das Programm nicht bildschirmorientiert ist (maximal 640x400 Punkte), kann der Benutzer die meisten Funktionen auf die gesamte Grafikseite anwenden. Bei einer maximalen Auflösung von 400 Punkten pro Zoll lassen sich so pro Seite bis zu 14,72 Millionen (!) Bildpunkte einzeln bearbeiten! Dies entspricht einem Seitenformat von maximal 4600 x 3200 Punkten. Zur besseren Übersicht stehen drei verschiedene Anzeigemöglichkeiten zur Auswahl. Bei der 1:1-Darstellung ist zwar immer nur ein kleiner Ausschnitt sichtbar, dafür arbeiten alle Funktionen sofort punktorientiert. Zudem ist eine UNDO-Funktion, die ungewollte Zeichenaktionen rückgängig machen kann, bei der 1:1-Darstellung fast uneingeschränkt möglich. Durch die hohe Anzahl an Bildpunkten und den damit verbundenen Speicherbedarf einer Seite ist ein UNDO jedoch nicht immer zu realisieren. Hier wird im Handbuch bei jeder Funktionsbeschreibung erwähnt, ob und in welcher Darstellungsart dies durchführbar ist. Ebenso verhält es sich mit der Beweglichkeit von Objekten nach dem Zeichnen.

Aus Geschwindigkeitsgründen stehen dem Benutzer drei feste Vergrößerungen zur Auswahl: Es kann im Modus 1:1, 1:8 oder 1:16 gearbeitet werden. Im Gegensatz zu Calamus wird erst dann in die neue Anzeigeart umgeschaltet, wenn eine Zeichenfunktion benutzt wird. Dies vermeidet unnötigen Seitenaufbau und spart Zeit. Zusätzlich erscheint in den Modi 1:8 und 1:16 ein sogenanntes „Super-Zoom-Out-Fenster". Hierin wird punktgenau der aktuelle Ausschnitt 1:1 gezeigt, an dem sich der Mauspfeil befindet. Dieser bewegt sich - je nach Auflösung - um 8 oder 16 Punkte. Auf Tastendruck kann jedoch in punktgenaues Arbeiten umgeschaltet werden. So läßt sich zum Beispiel ein Rechteck über eine ganze DIN A4-Seite sehr exakt setzen. Doch es lassen sich nicht nur Rechtecke oder Kreise über die komplette Grafikseite zeichnen. Auch Füllfunktionen und ähnliches erstrecken sich auf Wunsch ganzseitig.

An Scanner wurde in Paint natürlich auch gedacht. Drei Treiber stehen zur Verfügung - und zwar für SPAT-Flachbett-, HAWK CP 14-Flachbett- und den Handyscanner. Somit lassen sich auch Seiten einscannen, ohne die Scannersoftware aufrufen zu müssen. Ein im Test

passung an Atari TT und Großbildschirme aber bereits geplant.

Zukunftsperspektiven

Insgesamt ist die Shell mit Icons im Stil von Calamus ansprechend gelöst, hat vermutlich aber wenig Zukunft. Um als vielverwendete Shell eingesetzt werden zu können, müßten sich andere Software-Hersteller an Graffiti anpassen. Dies ist jedoch kaum wahrscheinlich, da die Shell in ihrem Aufbau zu unflexibel ist. Ihr Einsatz ist an und für sich nur im grafischen Bereich zu suchen. Insofern werden wohl andere Firmen kaum auf eine Graffiti-Programmierung umsteigen. Dagegen spricht auch die bisher noch sehr geringe Verbreitung von Graffiti. Man kann also davon ausgehen, daß nur die Firma K&L-Datentechnik das Graffiti-Konzept mit Modulen weiterhin unterstützen, erweitern und aufrechterhalten wird.

Handbuch

Das Handbuch geht auf 73 Seiten sehr ausführlich auf fast alle Programmpunkte ein. Mitunter stimmen die Angaben des Handbuches jedoch nicht ganz mit der tatsächlichen Programmbedienung überein. So heißt es etwa, daß zum Zeichnen einer Linie die Maustaste zum Festlegen des Endpunktes gedrückt gehalten werden muß, in Wirklichkeit sind jedoch zwei Mausklicks (einer für Anfang und einer für Ende) notwendig. Meist handelt es sich jedoch um Kleinigkeiten, die beim Umgang mit dem Programm schnell klar werden.

Um dem Benutzer den Einstieg in Graffiti zu erleichtern, wurde das Handbuch zahlreich bebildert. Wohl, um es nicht zu lang werden zu lassen, sind die Abbildungen oft so klein geraten, daß sie schlecht zu erkennen sind. Dies wird durch ein ausführliches Sachwortverzeichnis am Ende des Handbuches jedoch wieder wettgemacht. So fällt auch das Nachschlagen einzelner Funktionen nicht schwer.

Paint

Das Modul Paint ist ein vollständiges Zeichenprogramm. Daß Graffiti aus den Händen von Konstantinos Lavassas und Thomas Klingelhöfer stammt, läßt erwarten, daß sich einige Programmfunktionen aus ihrem Vorgängerprodukt „Lavadraw" wiederfinden. Hierauf muß der Käufer nicht lange warten - schon beim ersten „Reinschnuppern" in die Untermenüs fallen zahlreiche Möglichkeiten auf, die Lavadraw mehr als ähneln.

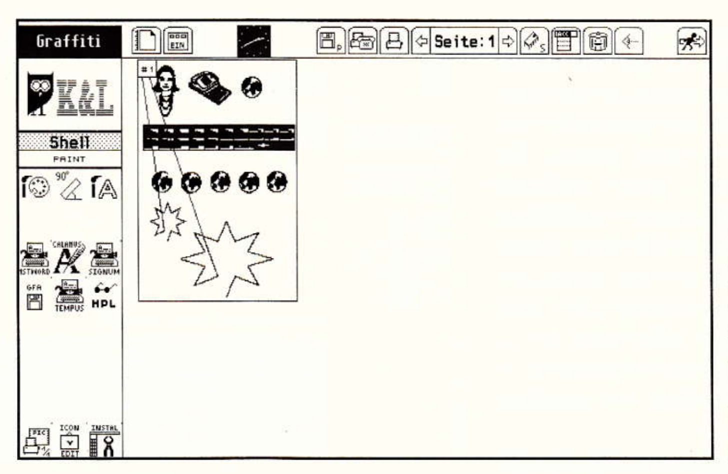


Bild 1: Graffiti dient als Shell und kann andere Programme auf Knopfdruck starten. Manche Zeichenfunktionen (zum Beispiel „Stern") sind jedoch noch nicht ganz fehlerfrei.

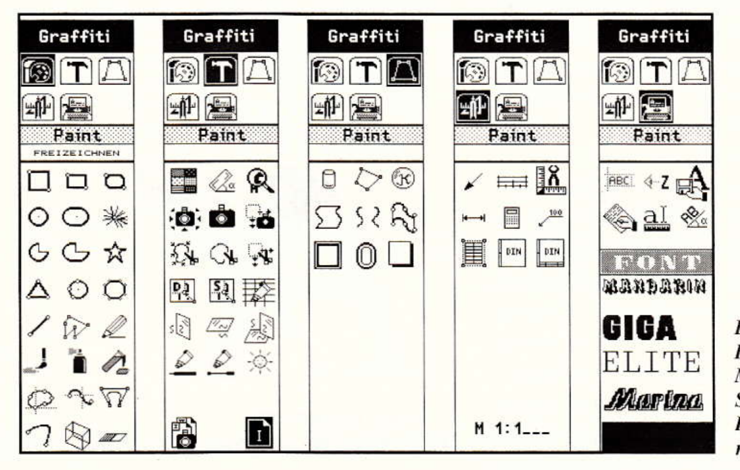


Bild 2: Im Modul Paint wurde das Menüsystem vieler Symbole dem DTP-Programm Calamus nachempfunden.

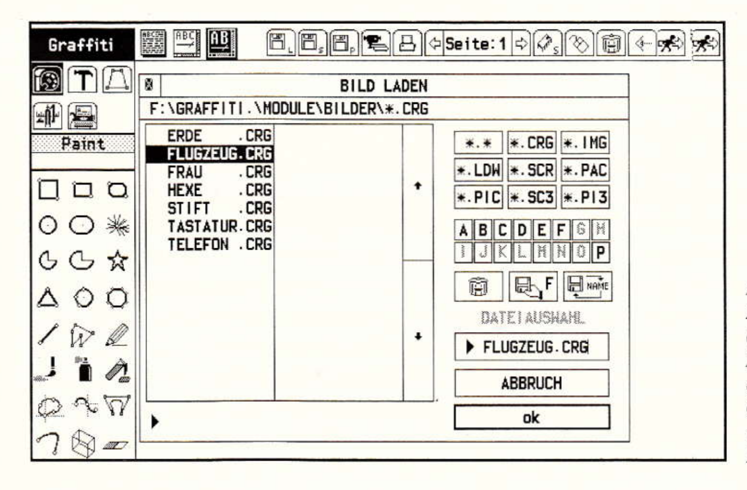


Bild 3: Die neue Dateiauswahlbox (Fileselector) gibt die Dateieindungen einladbarer Grafikstandards vor und kann bis zu 30 Einträge anzeigen.

Ganz anders dagegen ist die Menüsteuerung ausgefallen. Wie oben erwähnt, haben sich die Programmierer am Menüsystem der DTP-Software Calamus orientiert. In gleicher Art und Weise finden sich hier in verschiedenen Hierarchie-Ebenen Haupt- und Untermenüs wieder (siehe Bild 2). Dieses „etwas andere" Menü ist im Gegensatz zu „herkömmlichen" Menüleisten anfangs zwar ein wenig gewöhnungbedürftig, nach kurzem Arbeiten erweist es sich jedoch als sehr effektiv. Eine Referenzkarte, auf der die fünf Hauptmenüs mit ihren jeweiligen Funktionen abgebildet sind, wäre eine sehr nützliche Hilfe. Die Position der Menüeinträge ließe sich so noch leichter einprägen. Laut Handbuch wurde versucht, in allen Modulen gleiche Symbole für gleiche Funktionen zu verwenden. Leider trifft dies mitunter nicht auf die Einheitlichkeit in der Bedienung zu. Ungereimtheiten gibt es zum Beispiel, weil das Verlassen des Programmes mal mit der rechten Maustaste bestätigt werden muß, dann aber wieder mit einem Klick in eine Dialogbox.

Neben den üblichen Standardfunktionen bietet Graffiti mehrere Kopiermöglichkeiten und Effekte. Weiterhin stehen mehrere Texteingabearten für ein- oder mehrzeiligen sowie gedrehten Text zur Verfügung. Im Speicher lassen sich bis zu fünf Zeichensätze halten. Im Editor können per Tastenkombination alle ASCII-Zeichen eingegeben werden. Folglich auch solche, die sonst auf der Tastatur nicht

verwendeter Silverreed SPAT funktionierte als Scanner problemlos. Wenn es gelingt, in den Rechner eingelesene Bilder perfekt auf Papier zu bringen, kommt Freude auf. Hierzu existieren sieben feste Anpassungen: HP-Laser mit 150 und 300 dpi, Atari-Laser mit 300 dpi, Epson LQs mit einfacher oder doppelter Dichte bei 180 dpi oder mit hoher Dichte bei 360 dpi. Zudem wurde auch an Epson FX-Drucker (richtig, das sind die guten, alten „9-Nadler"!) mit 240 dpi gedacht. Das begeistert.

Als letzte Möglichkeit kann scheinbar ein externer Druckertreiber geladen werden. Wie dies geschehen muß, verschweigt das Handbuch. Zudem ist keine Angabe zu finden, wie der Treiber aufgebaut sein muß. Wer also einen Drucker sein eigen nennt, der die üblichen ESC/P- oder Laser-Druckkommandos nicht versteht, kann unter Umständen Probleme bekommen. Dafür gibt es Punktabzug.

Mit Graffiti lassen sich nur ganze Seiten ausdrucken. Laut Handbuch gibt es ein Modul zum Teilausdruck, was jedoch zusätzlich erworben werden muß. Ein weiteres Symbol ist in Graffiti enthalten, das bisher jedoch nicht belegt ist - ein Telefax-Gerät. Hier soll künftig ein Fax-Treiber integriert werden, der die eingeladene Grafikseite an ein Fax-Modem ausgibt. Sollte diese Option demnächst verwirklicht werden, könnte Graffiti im Telefax-Bereich zu einer interessanten Hilfe werden.

Zeichensatzeditor

Als angenehme Zugabe ist im Programmpaket ein Zeichensatzeditor enthalten, mit dem sogenannte Pixel-Fonts (Zeichensätze, die eine feste Größe haben und aus Punkten, nicht aus Linien, bestehen) bearbeitet werden können (siehe Bild 4). Dennoch ist der Zeichensatzeditor eher sehr spartanisch als luxuriös mit Zeichenfunktionen ausgestattet. Verändern von Buchstaben wird durch fehlende Grundfunktionen (Linien, Rechtecke, Radiergummi etc.) zum mühsamen, punktweisen Unterfangen. Dafür kann der Zeichensatzeditor mehrere Zeichensätze verarbeiten: Neben den eigenen Zeichensätzen liest er Signum!-Drucker-Fonts in allen drei Auflösungen (*.P9, *.P24 und *.L30) sowie Systemzeichensätze. Auf einer zweiten Diskette werden zahlreiche Zeichensätze im eigenen Format mitgeliefert, die großteils recht gut gelungen sind. Leider stimmt manchmal die Höhe der einzelnen Zeichen untereinander nicht exakt überein.

Um mit den beiden Programmdisketten gleichzeitig arbeiten zu können, ist es ratsam, wenigstens zwei Diskettenlaufwerke oder besser noch eine Festplatte zu besitzen. Sonst entwickelt sich das Hin- und Herjonglieren zweier Disketten zum ermüdenden Kampf. Der Einsatz einer Shell ohne Festplatte scheint ohnehin wenig sinnvoll. Zwar können die einzelnen Module in einem beliebigen Pfad installiert werden, sie sind jedoch ziemlich starr auf das vorgegebene Ordnersystem der Originaldiskette festgelegt. Versucht man etwa, alle Module aus ihrem eigenen Ordner (ein Modul = ein Ordner) in einen Sammelordner zu legen, finden manche Module trotz Umstellung der Pfadnamen ihre Dateien nicht mehr. Nach Möglichkeit sollten sämtliche Programme deshalb so in einem Ordner angelegt werden, wie auf der Originaldiskette vorgegeben. Das Ordnerproblem (viele Ordner, schlechte Übersicht) läßt sich also nur mit extremem Aufwand umgehen.



Icon-Editor

Mit dem Modul IconEdit können eigene Symbole erstellt werden. Dies ist jedoch wegen mangelnder Zeichenfunktionen (Linien, Rechtecke) ebenso mühsam wie die Änderung von Buchstaben im Zeichensatzeditor. Möchte ein Benutzer also tatsächlich seine am häufigsten benutzten Programme mit dieser Shell aufrufen (was sowieso nur mit Festplatte sinnvoll erscheint), müßte zunächst für jedes Programm ein Icon entworfen werden.

Fazit

Graffitti muß man in die zwei Teile „Shell" und „Grafikprogramm" unterteilen und diese getrennt beurteilen. Als Shell ist das Programm zwar ansprechend gelungen, wird dennoch kaum Zukunft finden und es schwer haben, sich durchsetzen. Die Folge wird sein, daß weitere Module wohl nur noch durch die Programmierer von Graffiti entwickelt werden. Dies erschwert den Einsatz als Shell für den täglichen Gebrauch. Diesen Programmteil sollte man eher als nützliche Beigabe zum Grafikprogramm Paint sehen.

Die Zeichen-Software bietet im Gegensatz zu vielen bildschirmorientierten Programmen die Möglichkeit, auch ganzseitig mit den Zeichenfunktionen zu arbeiten. Durch die ideale Kombination von Übersichtlichkeit in verringerter Darstellungsauflösung mit dem sogenannten Super-Zoom-Out-Fenster ist zwar seitenweises, aber trotzdem punktweises Arbeiten möglich. Zahlreiche Funktionen lassen Freude beim Bearbeiten von DIN A4-Seiten aufkommen. Für den Mindestspeicherbedarf von zwei Megabyte ist nicht das Programm Paint, sondern die hohe Auflösung verantwortlich. Die Vielseitigkeit beim Laden und zudem auch beim Abspeichern von Grafikfremdformaten überzeugt sofort alle die Zeichner, die mit mehreren anderen Programmen arbeiten.

Der Zeichensatzeditor ist etwas spartanisch ausgefallen. Ein paar Zeichengrundfunktionen wären hier dringend nötig. Dies wird durch die Zugriffsmöglichkeit auf die große Zahl an Signum!-Zeichensätzen halbwegs ausgeglichen. Der Icon-Editor in der Minimal-Serienausstattung genügt gerade, um Punkte zu setzen oder zu löschen. Es stehen nur die notwendigsten Funktionen für die Symbolerstellung zur Verfügung. Auch hier wäre etwas mehr Komfort wünschenswert.

Höchst interessant wird Graffiti, wenn die angekündigte Funktion zum Senden einer kompletten Grafikseite über ein Fax-Modem fertiggestellt sein wird.

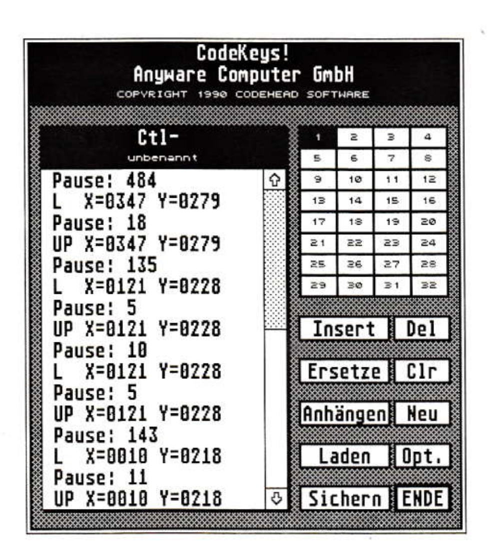
Alles in allem bietet Graffiti heute noch wenig neue Funktionen, die sich deutlich von auf dem Markt eingeführten Produkten der Mitbewerber unterscheiden. Größter Vorteil ist, daß die meisten Funktionen auch auf einer ganzen Bildschirmseite angewandt werden können. Derjenige, der nach wie vor mit 640 x 400 Bildpunkten zufrieden ist, wird die Anschaffung mehrfach überdenken und dabei sicher auch wegen des Preises von DM 349,- nicht zuletzt auf das Erscheinen der Telefax-Funktion warten.

RP

Bezugsquelle:
K&L-Datentechnik
Bahnhofstr. 11
W-3551 Bad Endbach
Telefon (02776) 8145

CodeKeys

Der Makro-Manager



Bislang verfügen nur wenige Programme über die Möglichkeit, Makros zu definieren, über die mehrere Operationen mit einem einzigen Tastendruck aufgerufen werden können. Durchaus verständlich ist der Wunsch des Anwenders, möglichst aus jeder Situation heraus vordefinierte Aktionen zu starten. Das Programm CodeKeys greift hier unter die Arme.

Seit dem Erscheinen von MegaSTE und TT ist es immerhin möglich, die Desktop-Funktionen über die Tastatur aufzurufen und so nicht für kleinere Aktionen ständig zur Maus greifen zu müssen. Atari hat also Einsicht gezeigt und einen häufig geäußerten Wunsch der Anwender endlich in die Tat umgesetzt.

Zur Zeit bieten in erster Linie Textverarbeitungen sogenannte Makros, die es erlauben, auch komplexe Operationen zu automatisieren. Allgemein einsetzen lassen sich solche Makros natürlich nicht, sie sind lediglich für den Einsatz in einem ganz bestimmten Programm geeignet.

CodeKeys erlaubt es nun, Makros aus jeden Programm heraus zu definieren und aufzurufen. Alle Operationen des Benutzers, zu denen auch Mausaktionen zählen, können aufgezeichnet und zu einem späteren Zeitpunkt wieder abgespielt werden.

Vierfinger-System

Die Bedienung von CodeKeys kann auf zwei Arten erfolgen. Zum einem kann man Makros über das mitgelieferte Accessory definieren, zum anderem läßt sich das Programm in einigen Funktionen auch über Tastenkombinationen steuern.

Zunächst zur letztgenannten Möglichkeit. Vielleicht sind Sie ja schon im Besitz eines Programms, dessen Bedienung sich dadurch hervorhebt, daß man zur Aktivierung gewisser Funktionen mehrere Shift-Tasten gleichzeitig betätigen muß. Je mehr solcher Programme auf den Markt kommen, umso einfallsreicher werden die Tastenkombinationen, mit denen die Bedienung erfolgen soll. Schließlich gilt es, Doppeldeutigkeiten zu vermeiden. CodeKeys macht hier keine Ausnahme. Vier Finger sollten Sie schon frei haben, um das Programm über die Tastatur zur Aufnahme eines Makros zu veranlassen. Daß es auch eine Funktion gibt, die über fünf gleichzeitig zu betätigende Tasten aufgerufen werden kann, sei hier nur am Rande erwähnt.

Glücklicherweise haben die Entwickler von CodeKeys es nicht dabei belassen, den Anwender ausschließlich mit Fingerakrobatik zu belasten. Alle Funktionen von CodeKeys können über ein Accessory gesteuert werden, so daß die Tastaturbedienung höchstens dann notwendig wird, wenn man sich in einem Programm befindet, das den Zugang zur Menüleiste verwehrt. Programme dieser Art findet man inzwischen aber kaum noch.

Achtung, Aufnahme

Schauen wir uns näher an, wie CodeKeys arbeitet. Am Anfang steht die Definition eines Makros, die anhand einer Rekorder-Funktion erfolgt. Nachdem die Taste bestimmt wurde, durch die das Makro später ausgelöst werden kann, merkt sich das Programm alle folgenden Aktionen, egal, ob es sich um Tastatureingaben oder Mausklicks handelt.

Bis zu 32 Makros mit jeweils 128 Befehlen können aufgenommen werden. Als Befehle sind dabei auch Pausen zwischen den einzelnen Kommandos anzusehen. Zwar erlaubt es CodeKeys, Makros mit maximaler Geschwindigkeit auszuführen, allerdings dürften in den meisten Fällen Pausen zwischen den Aktionen notwendig sein. Andernfalls besteht die Gefahr, daß nicht genügend Zeit zur Befehlsausführung bleibt und bereits versucht wird, das nächste Makro-Kommando auszuführen, bevor das letzte beendet wurde. Berücksichtigt man die Pausen (Aufnahme in Echzeit), so bleiben unterm Strich noch 64 echte Befehle für ein einzelnes Makro. Allzu komplexe Funktionen sind so nicht möglich. Es besteht jedoch die Möglichkeit, daß ein Makro ein anderes Makro aufruft, so daß sich längere Befehlssequenzen über diesen kleinen Umweg generieren lassen.

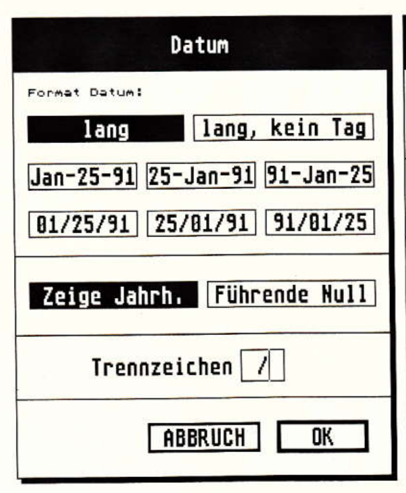
Makros können übrigens auf jede beliebige Taste gelegt werden. Auch Kombinationen in Verbindung mit den Shift-Tasten sind möglich.

Makro-Editor inklusive

Nun kann es vorkommen, daß einem während der Makro-Aufnahme ein Fehler der Art unterläuft, daß man eine Operation startet, die eigentlich gar kein Bestandteil des Makros sein sollte. In solchen Fällen braucht man sich nicht darüber zu ärgern, die gesamte Makro-Definition wiederholen zu müssen. CodeKeys besitzt nämlich einen Makro-Editor, der es erlaubt, bereits definierte Makros nachträglich zu edieren. So bereitet es keine Schwierigkeiten, ein Makro zu erweitern oder um bestimmte Befehle zu kürzen. Darüber hinaus ist es

möglich, ein Makro zu duplizieren, so daß es ein leichtes ist, aus bereits bestehenden Makros durch leichte Abwandlungen neue Befehlssequenzen zu erhalten.

Wurde ein Makro in Echtzeit (also mit Pausen) aufgenommen, so erlaubt es der Makro-Editor, die Dauer der Pausen zu verkürzen, so daß das Abspielen des Makros mit optimaler Geschwindigkeit erfolgen kann. Solche Änderungen stellen jedoch kein Muß dar, sie ermöglichen es lediglich, die Befehlsbearbeitung zu beschleunigen.



Zeit und Datum in Makros

Spezielle Makro-Funktionen

CodeKeys beläßt es nicht dabei, lediglich Tastatur- und Mausaktionen in einem Makro unterbringen zu können. Es existieren zusätzlich einige Spezialfunktionen, die den Einsatzbereich von Makros erweitern. So ist es möglich, Zeit und Datum in verschiedenen Formaten in ein Makro einzufügen. Sehr praktisch ist diese Funktion vor allen Dingen dann, wenn es um das Schreiben von Briefen geht. Per Tastendruck kann man mit einem geeigneten Makro ganze Teile des Briefkopfes inklusive Datum in den Text einfügen.

Sonderzeichen des Atari-Zeichensatzes, die nicht über die Tastatur erreichbar sind, können nachträglich mit Hilfe des CodeKeys-Accessories in eine Makro-Definition eingefügt werden. Der Zugriff auf alle auf dem ST/TT zur Verfügung stehenden Zeichen ist also sichergestellt.

Problemfall Maus

Die Bedienung von CodeKeys gestaltet sich im großen und ganzen unkompliziert. Beim Aufzeichnen von Mausaktionen ist jedoch eine gewisse Vorsicht geboten. Es ist nämlich nicht möglich, Mausbewegungen aufzuzeichnen. Dies scheint zunächst nicht weiter von Bedeutung zu sein, kommt aber dann zum Tragen, wenn man versucht, mehr als ein Objekt per Maus zu selektieren. CodeKeys erkennt in diesem Fall nicht, wenn mit Hilfe der Maus ein Lasso aufgespannt wird, mit dem mehrere Objekte gleichzeitig erfaßt werden sollen.

Geht es darum, Objekte innerhalb eines Fensters auszuwählen, ist unbedingt darauf zu achten, daß sich das Fenster beim Abspielen des Makros an der gleichen Stelle befindet wie bei der Makro-Definition. Beachtet man dies nicht, so werden keine oder die falschen Objekte angesprochen.

Ähnlich sieht es aus, wenn ein Makro nicht in der Bildschirmauflösung aufgenommen wurde, in der es abgespielt wird. Auch hier kann es Probleme geben, falls sich die Bildschirmkoordinaten für Mausaktionen nicht entsprechen. CodeKeys erlaubt es jedoch, auflösungsabhängige Makro-Dateien zu erstellen, so daß für jede Auflösung ein eigener Satz an Makros zur Verfügung steht. Gerade für Besitzer eines Atari TT ist diese Möglichkeit wichtig, da hier das Arbeiten in mehr als einer Auflösung durchaus üblich ist.

Um Schwierigkeiten bei der Aufnahme von Mausaktionen aus dem Wege zu gehen, sollte man, soweit möglich, beim Definieren von Makros zugunsten von Tastaturkommandos auf die Maus verzichten.

Automatik

Makros lassen sich nicht nur über die Tastatur aktivieren. Die Entwickler von CodeKeys haben noch einige zusätzliche Möglichkeiten vorgesehen, mit denen ein Makro aufgerufen werden kann.

Zunächst einmal können Makros zu einem vordefinierten Zeitpunkt gestartet werden. So läßt sich beispielsweise eine Wecker-Funktion realisieren, wobei natürlich auch sinnvollere Anwendungen denkbar sind. Das CodeKeys-Handbuch schlägt hier eine Backup-Funktion per Makro vor. Wenn es um das Backup einzelner Dateien geht, die vom Desktop aus markiert werden, dürfte man jedoch Probleme bekommen. Das Markieren mehrerer Objekte mit der Maus ist schließlich, wie bereits angesprochen, nicht ohne weiteres möglich.

Besonders interessant sind Makros, die periodisch aufgerufen werden. Arbeitet man mit einer Textverarbeitung, so könnte man anhand eines solchen Makros dafür sorgen, daß nach einer gewissen Zeitspanne der aktuelle Text gesichert wird, ohne daß man sich selber darum kümmern muß.

Schließlich gibt es noch die sogenannten „Autorun-Makros". Dabei handelt es sich um Makros, die nach dem Start eines Programms automatisch ausgeführt werden. Damit erhält man die Möglichkeit, direkt nach dem Aufruf des Programms diverse Aktionen ausführen zu lassen, die dazu dienen können, Voreinstellungen zu aktivieren, die man andernfalls per Hand durchführen müßte.

Es ist übrigens möglich, für ein Programm einen eigenen Satz an Makros bereitzuhalten. Diese werden beim Programmstart automatisch aktiviert und stellen somit für jedes Programm eine ganz bestimmte Arbeitsumgebung zur Verfügung.

Blick ins Handbuch

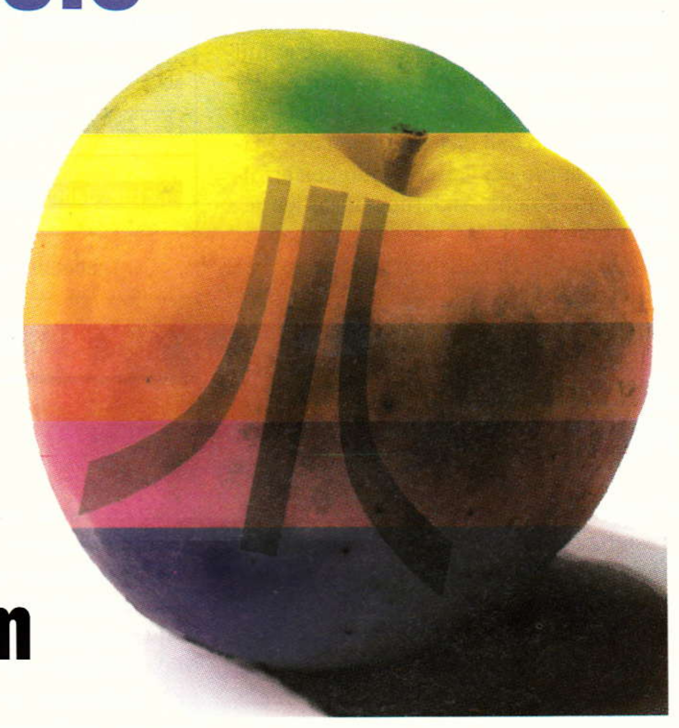
Die Programmbeschreibung zu CodeKeys kann durchaus als gelungen bezeichnet werden. Ausführlich und locker werden die Möglichkeiten zum Einsatz des Programms erläutert, ohne daß der Leser überfordert wird. Piktogramme weisen auf besonders wichtige Informationen hin, alle Operationen werden durch Bilder erläutert. Auch der weniger erfahrene Atari-Anwender wird nach dem Studium der Anleitung in der Lage sein, CodeKeys effektiv einzusetzen.

Sowohl das Programm als auch das Handbuch zu CodeKeys hinterließen bei mir einen überwiegend positiven Eindruck. Für 98 DM erhält man ein Programm, das den Umgang mit Atari ST und TT in vielen Situationen erleichtern kann. Lediglich bei Mausoperationen lassen sich Makros nicht so vielseitig einsetzen, wie es wünschenswert wäre.

US

Bezugsadresse:
Anyware Computer GmbH
Holbeinstraße 60
W-6000 Frankfurt 70
Tel. (069)6312456

Spectre 3.0



...und der Apfel fällt ganz weit vom Stamm

Den letzten Bericht über den Spectre, den Emulator, der den Atari-Rechnern das Leben eines Apple Macintosh einhaucht, konnten Sie bei uns vor einiger Zeit lesen (genau genommen in der Mai-Ausgabe '89). Seitdem hat sich einiges getan. Zunächst einmal ist die deutsche Konkurrenz des Spectre, Aladin, aufgrund von Rechtsstreitigkeiten mit Apple vom Markt verschwunden. Es hat sich mal wieder gezeigt, daß Apple nicht sehr gut auf Clones und Emulatoren zu sprechen ist. Doch der Spectre scheint damit bis jetzt noch keine Probleme zu haben. Er liegt mittlerweile in der Version 3.0 vor und läuft auf ST, STE und TT.

Den Spectre gibt es in zwei Ausführungen (128 und GCR), wobei die interessanteste mit Sicherheit die GCR-Version ist, mit der man mit Atari-Laufwerken auch Mac-Disketten lesen und schreiben kann. GCR steht einfach für Group Code Recording. Damit ist das Diskettenaufzeichnungsverfahren des Mac gemeint, mit dem man bei Apple im Gegensatz zu Atari- oder IBM-Rechnern ein eigenes Süppchen gekocht hat. Es soll hier aber nicht näher auf dieses Format eingegangen werden. Es sei nur soviel gesagt, daß die Spuren mittels Hardware mit unterschiedlicher Geschwindigkeit gefahren werden, um eine bessere Datendichte zu erreichen. Dadurch werden so beschriebene Disketten normalerweise für Mac-fremde Rechner unlesbar, da die nötige Hardware fehlt. Dem Spectre GCR ist aus diesem Grunde ein zusätzlicher Floppy-Controller spendiert worden, der dieses Wunderwerk zustandebringt.

Beide Spectre-Versionen benötigen mindestens 1 MB RAM, Englischkenntnisse für das lesenswerte Handbuch und werden in den ROM-Port gesteckt. Zum Test stand uns ein Spectre GCR zur Verfügung, der zusätzlich über Floppy-Hardware verfügt und mit einem Floppy-Kabel an den Atari angeschlossen werden muß. Dazu finden sich am GCR zwei Buchsen, von denen die zweite zum Anschluß eines optionalen externen Laufwerks dient. Das mitgelieferte Floppy-Kabel ist derzeit zum Anschluß des GCR an einen Mega STE noch etwas zu kurz, was aber mit Sicherheit in Kürze behoben sein wird.

Suche ROMs

Der Spectre benötigt zum Betrieb die 128k-ROMs eines Mac Plus. HG Computersysteme bietet diese zwar auch an, aber sie sind nicht jederzeit verfügbar. Dank den Argusaugen Apples ist es nicht gerade einfach, diese in ausreichenden Stückzahlen zu besorgen, denn man kann sie nicht einfach bei einem Apple-Handler kaufen. Apple will so kompatiblen Nachbauten der Macs vorbeugen, die dann ja auch ROMs benötigen. Nebenbei sei erwähnt, daß es bei Apple mittlerweile schon 256k- und in den neuesten Modellen 512k-ROMs gibt. Die 128k-ROMs sind also aus Apple-Sicht eigentlich ein alter Hut, für den Spectre aber lebensnotwendig. Die 3.0-Version des Spectre unterstützt übrigens nicht mehr die alten 64k-Mac-ROMs, die noch für Aladin und die ersten Spectre-Modelle benötigt wurden.

Doch ROMs sind noch nicht alles. Das Betriebssystem des Mac ist in zwei Teile aufgeteilt: ein Teil befindet sich im ROM, der andere auf Diskette bzw. Festplatte. Durch diese Aufteilung hat man eine sich bei Apple den Weg offengehalten, immer wieder verbesserte Versionen der Betriebssystem-Software herausgeben zu können. Dies ist ein riesiger Vorteil gegenüber Atari, wo System-Updates immer mit neuen ROMs verbunden sind.

Die neueste Version des Mac-Betriebssystems lautet 7.0 und soll angeblich auch schon auf dem Spectre laufen. Wir konnten das noch nicht testen, da diese Version z.Zt. nur Apple-Entwicklern zu-



Die Zukunft inbegriffen

Warum gleich einen TT kaufen und den alten ST in die Ecke stellen? Wenn Sie Geschwindigkeit brauchen, haben wir für Sie eine praktische und natürlich günstige Lösung: NVDI. Aber auch wenn Sie schon einen TT haben, gilt dieser Rat für Sie. Ihr TT wird so schnell, daß Sie ihn nicht mehr erkennen werden. NVDI ist die Lösung für viele, viele ATARI-Anwender, die sich schon immer eine schnellere Bildschirmausgabe gewünscht haben.

NVDI enthält ein vollständiges GDOS, wodurch das lästige Vorladen eines solchen Programmes entfällt. Sie bekommen somit summa summarum zwei Programme in einem.

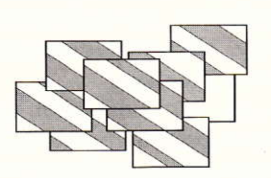
NVDI ist vielfältig und sehr anpassungsfähig. Es arbeitet mit vielen Beschleunigerkarten zusammen (z.B Board 20 von MAXON, HyperCache030 von ProVME). Auch unsauber programmierte Anwendungen behindern die Arbeit von NVDI nicht.

NVDI beschleunigt nicht nur den normalen Schwarzweiß-Modus des ATARI ST, sondern auch andere Auflösungen wie die von OverScan, MegaScreen, MAXON Graphic Adapter oder Matrix-Karte.

..und nicht nur das. Durch sein Konzept ist NVDI eine Lösung, in der die Zukunft schon inbegriffen ist. Nutzen auch Sie die Möglichkeit, der Zukunft einen Schritt voraus zu sein.

NVDI
Die Lösung
Unverbindliche Preisempfehlung DM 99.-

REVOLVER



Der Profi-Switcher für Ihren ATARI ST. Wo andere Programme den Dienst quittieren, da bietet REVOLVER Sicherheit. Resetfest in jedem Rechnerteil und mit umfangreichen Utility-Funktionen ist REVOLVER ideal für Programmierer, Musiker und Anwender, die mehr aus ihrem ATARI ST machen wollen.

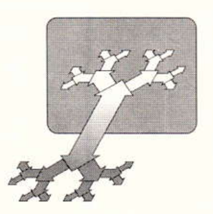
REVOLVER -
Der Profi-Switcher
Unverbindliche Preisempfehlung
DM 79,-

STOP

Einbruch und Datendiebstahl - kein Thema auf dem ST? Mit STop schützen Sie persönliche Daten, Programme oder Artikel- und Kundendateien vor fremden Zugriff. Nur über die Paßwörter ist der Echtzeitzugriff auf die vollständig kodierten Daten möglich. Die Datensicherheit dürfte mit 256 hoch 256 Möglichkeiten gewährleistet sein!

STOP -
Der Datentresor
Unverbindliche Preisempfehlung
DM 129,-

XBoot



XBoot jetzt in der Version 2.5

XBoot, das Allround-Boot-Talent, präsentiert sich jetzt in der Version 2.5. Noch komfortabler, viel leistungsfähiger und vielfältiger denn je. **Für wen?**

Für **Designer,** die ständig DTP-Applikationen und Grafikprogramme benutzen müssen. XBoot installiert Ihren Rechner so, daß Sie sofort mit CALAMUS arbeiten können. Oder vielleicht mit ARABESQUE? In jedem Fall wird Ihr Rechner so gebootet, wie SIE es möchten.
Arbeiten Sie zufällig mit einem Groß- und einem Kleinbildschirm? Kein Problem, XBoot erledigt für Sie das lästige Installieren verschiedener Programme. Und das für immer. Für Sie als Designer, Grafiker, Layouter, Gestalter etc. ist XBoot also das ideale Komplement für Ihre tägliche Arbeit.

Für **Programmierer,** die immer und immer wieder auf verschiedene Konfigurationen zugreifen müssen, ist XBoot ein, pardon, das Utility für die tägliche Arbeit mit dem Computer. Eine winzige Zeile - und Sie haben Ihr komplettes Programmierwerkzeug in einer von Ihnen angemeldeten RAM-Disk.

Für **Anwender** im allgemeinen, die viele Accessories nachladen, ist XBoot die Hilfe schlechthin. Sie brauchen XBoot nur mitzuteilen, welches Accessory Sie im Moment brauchen. Den Rest erledigt XBoot von selbst. Möchten Sie, daß während Ihrer Abwesenheit niemand an Ihren Rechner herankommt? Auch das erledigt XBoot in der Version 2.5. Ein Paßwort verhindert den Zugriff Unbefugter.

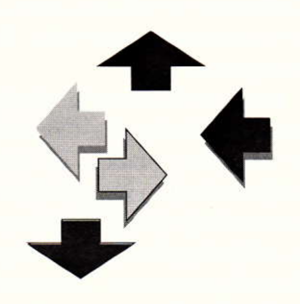
Und wie Sie wissen, alles hat seinen Preis. XBoot ebenso. Aber für **DM 79.-** hat XBoot eine Menge zu bieten.

SALDO

SALDO ist ein Programm, das Ihnen erlaubt, die Kontrolle Ihrer Finanzen in den Griff zu bekommen. Sie können SALDO für private Zwecke, aber genausogut als Einnahme- und Überschußverwaltung für die gewerbliche Tätigkeit einsetzen. SALDO bietet mit seiner Vielzahl an Funktionen alle nur denkbaren Möglichkeiten, die eingegebenen Daten zu manipulieren. Sie können z.B. sortiert oder aufgesplittert nach verschiedenen Kriterien auf dem Bildschirm dargestellt oder auf dem Drucker ausgegeben werden.
Es würde einfach zuviel, hier jedes einzelne Detail von SALDO aufzuzählen - man muß es gesehen haben.

SALDO
Unverbindliche Preisempfehlung
DM 79.-

INTERLINK ST



INTERLINK ST ist das komfortabelste DFÜ-Programm für den ATARI ST und damit ideal für den Einsteiger und den Profi. So urteilen zumindest die Besitzer, die die Kommunikation und den weltweiten Datenaustausch mit Hilfe von INTERLINK ST nicht mehr missen möchten. Wann gehen Sie auf die Datenreise?

INTERLINK ST -
DFÜ im Griff
Unverbindliche Preisempfehlung DM 79,-



BELA Computer Layout- und Vertriebs GmbH Unterortstr. 23-25 W-6236 Eschborn Tel: 06196/481944 Fax: 481930

gänglich ist und erst in Kürze auf den Markt kommt. Bis jetzt muß man sich noch mit der Version 6.0.7 begnügen.

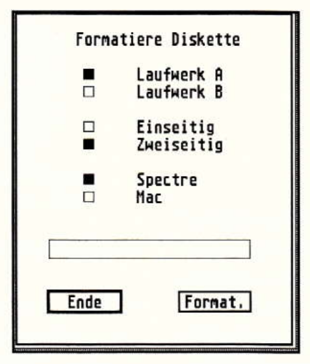
Nachhilfe für Laufwerke

Nach dem Anschluß des GCR stellte es sich bald heraus, daß einige Atari-Laufwerke in unserer Redaktion nicht gerade sehr Spectre-freundlich waren und teilweise Probleme bereiteten. Das variiert je nach Rechner, da nicht immer die hochwertigsten Laufwerkstypen in die Ataris eingebaut sind. „Power without the price" setzt nun mal auch voraus, daß man günstig am Markt einkauft. Zum Betrieb des Spectre wird aber auf jeden Fall ein einwandfrei justiertes Laufwerk benötigt, so daß zu empfehlen ist, schon beim Kauf auf einwandfreies Funktionieren zu achten. Hat man einen Spectre gekauft und Probleme mit seinem Laufwerk, wird man aber nicht im Regen stehen gelassen. Von HG Computersysteme werden modifizierte externe Laufwerke oder eine preiswerte Umrüstung von NEC-Laufwerken (1035, 1036 und 1037) angeboten. Der Spectre liest und schreibt übrigens auch Aladin-Disketten, so daß man auch auf Software des „ausgestorbenen" Mac-Emulators zurückgreifen kann.

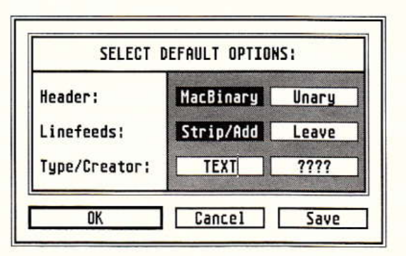
Prinzipiell sollte ein Mac bzw. Spectre sowieso mit einer Fest- oder Wechselplatte betrieben werden, da man ansonsten aufgrund des zweigeteilten Betriebssystems leicht zum „Diskjockey" wird, also so ziemlich bei jedem Zugriff auf das Betriebssystem die Systemdiskette einlegen muß. Will man dies vermeiden und ist stolzer Besitzer einer Platte, sollte man sich eine oder mehrere Partitionen darauf für den Spectre-Betrieb reservieren. Es kann eine Boot-Partition für das Mac-System angelegt werden, so daß man schnell die Systemdisketten vergessen kann. Spectre unterstützt übrigens bis zu 16 Partitionen.

Ein Tip am Rande: Man sollte immer die letzte Partition für den Spectre auswählen, da die Partitionskennung vom ihm von GEM bzw. BGM auf OOP geändert wird. Das hat zur Folge, daß diese Partition nicht mehr vom TOS erkannt wird. Nimmt man eine andere (z.B. die zweite) Partition für den Spectre-Betrieb, ändern sich aus diesem Grund die Laufwerksbezeichnungen der nachfolgenden Partitionen, und es müssen somit alle Pfadnamen anderer Programme neu eingestellt werden.

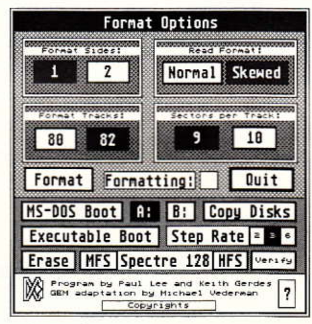
Der Spectre arbeitet mit allen ACSI- und SCSI-Platten zusammen. Ausnahme bildet hier derzeit noch die interne SCSI-Platte des TT bzw. Mega STE, was laut Hersteller in der Version 3.1 aber behoben sein soll. Hier hat Atari einem mal wieder



Die Formatbox des Spectre-Installationsprogramms



Mit dem Transverter lassen sich Daten zwischen der Mac- und der Atari-Welt übertragen.



Ein umfangreiches Formatprogramm wird ebenfalls mitgeliefert.

einen Streich gespielt und eine SCSI-Kennung größer 7 verwendet, die der Spectre momentan noch nicht erkennt. Keine Probleme bereiten allerdings externe ACSI-Platten an diesen Rechnern.

Vom Distributor werden SCSI-Platten verschiedener Größe z.B. mit Quantum(19 ms)- oder Fujitsu(20 ms)-Laufwerken angeboten, die sich unter Spectre, TOS und sogar am Mac und PC betreiben lassen. Spectre kann z.B. mit diesen Laufwerken original-Mac-formatierte und bespielte Platten lesen und beschreiben und umgekehrt. Dies vereinfacht den Datenaustausch deutlich, da man den Umweg über Diskette sparen kann. Für den ST-Betrieb wird ein ICD-SCSI-Interface mit entsprechender Treiber-Software mitgeliefert. Die angebotenen Platten passen vom Design übrigens ideal zu TT und Mega STE, was aber zufällig ist, da sie eigentlich auf das Mac-Design abgestimmt sind und normalerweise für diesen angeboten werden. Entsprechend haben sie auch eine auf den Mac angepaßte Anleitung und zusätzlich Mac-Software. Als ideal für den gemischten Betrieb dürfte sich ein Wechselplattenlaufwerk erweisen, da man hier sowohl Medien für ST- als auch Mac-Betrieb nutzen kann und sich keine Sorgen um die Partitionierung zu machen braucht. Die Preise fangen bei DM 1180,- für eine 40-MB-Festplatte an.

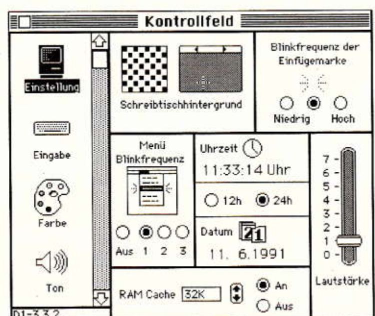
Startup

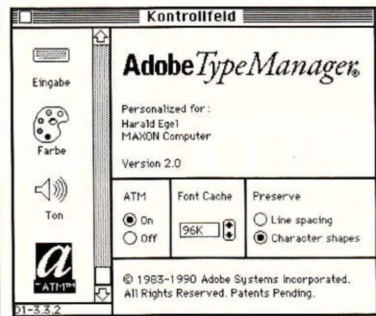
Nach diesem ausführlichen Ausflug kommen wir endlich zum Start des Spectre. Auf der Diskette befinden sich neben einem Programm, in dem alle Grundeinstellungen, Formatierungen etc. gemacht werden können, ein Startprogramm und diverse Ordner mit Zusatztreibern [z.B. für Tastatur (frei edierbar), *UltraScript* oder Betrieb eines PostScript-Laserdruckers am seriellen Port]. Nach dem Start wird man aufgefordert, eine Mac-Systemdiskette einzulegen (sofern man keine Boot-Partition angelegt hat). Schon bald findet man sich auf dem Mac-Desktop, dort Finder genannt, wieder und kann loslegen.

Im Gegensatz zu einem original Mac verfügt Spectre über jede Menge zusätzlicher Funktionen, die über Tastenkombinationen zu erreichen sind. So finden sich z.B. diverse Möglichkeiten für den Atari-Laserdrucker SLM 804 und ein Debug-Modus mit Diskmonitor. Ferner läßt sich der Spectre mit einem Farbmonitor betreiben, leider aber nur in Monochrom. Dabei handelt es sich mit Sicherheit um ein Zugeständnis an Länder, in denen der Atari fast nur in Farbe betrieben wird (USA, England etc.). Eine richtige Farbunterstützung ist derzeit leider nicht möglich, da beim Mac Farbe erst ab den 256k-ROMs Einzug gehalten hat. Vielleicht gibt es ja auch mal einen Spectre, der diese ROM-Version unterstützt. Nur werden diese noch schwieriger zu besorgen sein.

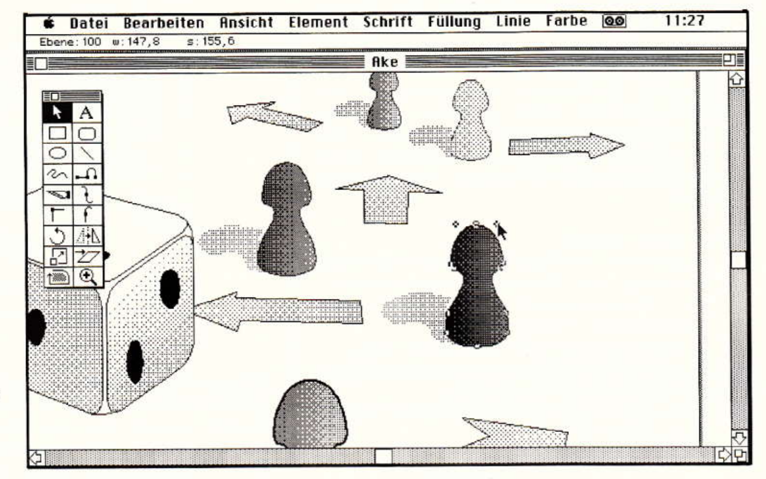
Der Spectre paßt sich automatisch der Bildschirmauflösung an. So arbeitet er z.B. auch mit Großbildschirmen und in der TT-hoch-Auflösung (1280x960 Pixel). Ein Overscan-Modus wird derzeit noch nicht unterstützt.

Auch wird ein vorhandener Arithmetik-Coprozessor für Mac-Programme genutzt. Der Prozessor kann sowohl in einem TT als auch in einer Beschleunigerkarte vorhanden sein. Laut Herstellerangaben wird die Atari-Erweiterung SFP004 nicht unterstützt, da sie auf eine „atari-spezifische Art" angesprochen wird. Damit ist sicherlich die Methode gemeint, den Coprozessorzugriff über einen Bus-Error zu handlen. Das ist bei Ataris mit 68000-Prozessor üblich, da die normale Methode über den Line-F-Emulator dort nicht möglich ist (er ist dort mit Grafikaufgaben





Der Mac verfügt über ein variables Kon-trollfeld, in dem man Para-metereinstellun-gen des Systems und von INITs (ähnlich Auto-Ordner-Pro-grammen) vor-nehmen kann.



Freehand bietet eine Vielzahl von grafischen Möglichkeiten, die durch Spectre nun auch auf dem Atari zur Verfügung stehen.



Ein Teil dieses Artikels in PageMaker 4.0 layoutet

betraut). Man dürfte also davon ausgehen können, daß alle Erweiterungen mit 68020/68030-Prozessoren funktionieren, da diese den Coprozessor über ein Busprotokoll ansprechen, das über den Line-F-Trap geregelt wird.

Ansonsten wird sich jeder, der schon einmal mit einem Mac gearbeitet hat, sofort zurechtfinden. Alles funktioniert wie gewohnt. Wir testeten Spectre mit diversen Mac-Programmen, wie z.B. *PageMaker 4.0*, *Norton Utilities*, *HyperCard 2.0* und *Freehand 2.02*, bei denen es keine Probleme gab. Freundlicherweise wird beim Spectre auch eine lange Kompatibilitätsliste mitgeliefert, in der sich jeder vorab informieren kann, ob seine Anwendung funktioniert. Auch Programme, die Probleme bereiten, sind mitaufgeführt. In der Regel handelt es sich dabei aber um sehr hardwarenahe Programme, die zum Teil auch nicht gerade der Apple-Norm entsprechen. Sollte der Spectre trotzdem mal ins Nirwana entschwinden, ist er um einiges ausführlicher als sein Vorbild. Während der Mac einfach eine Dialogbox mit der Fehlernummer (entspricht der Anzahl der Bomben beim Atari) ausgibt, erhält man beim Spectre neben einer Mitteilung über die Ursache des Fehlers auch noch die Registerinhalte des Prozessors zurück. Anschließend kann man zum Finder zurückkehren oder einen Neustart durchführen, wodurch geöffnete Dateien sauber abgeschlossen werden. Beim TT funktioniert das leider nicht, da dies eine resetfeste Installierung des Spectre voraussetzt, was bei der geänderten Speicherverwaltung des TT (ST- und TT-RAM) problematisch ist.

Ein Blick in die Liste zeigt übrigens, daß kopiergeschützte Software nicht läuft. Das ist darauf zurückzuführen, daß hier der Floppy-Controller des Mac meist direkt programmiert wird, was auch durch den zusätzlichen Controller des GCRs nicht zu emulieren ist.

Ab Systemversion 6.0 wurde beim Mac ein neuer Sound-Manager eingeführt, der bei älteren Spectre-Versionen zu Abstürzen führte. Das ist jetzt beim Spectre 3.0 behoben. Gerade eine so verbreitete Anwendung wie *Hypercard* bietet gute Sound-Möglichkeiten, denn dafür gibt es einen ziemlich großen PD-Pool von Anwendungen, die diese massiv nutzen.

Spectre macht Druck

Im Konfigurationsprogramm kann man einstellen, über welchen Port des Atari der Spectre drucken soll. Dabei stehen der serielle, parallele für normale und der DMA-Port für den Atari-Laserdrucker zur Verfügung. Früher war eine Hardcopy mit dem Spectre auf einem Atari-Laserdrukker nur in 72 dpi (Bildschirmauflösung) möglich, jetzt werden die vollen 300 dpi genutzt. Nur bei der um 90° gedrehten Hardcopy druckt er weiterhin mit 72 dpi. Der Spectre verfügt außerdem über eine eingebaute Apple ImageWriter-Emulation (72 dpi), die aber kaum hohe Ansprüche erfüllt. Will man bessere Qualität, ist man auf Druckertreiber von Fremdherstellern angewiesen. Hier werden von HG Computersysteme und anderen Firmen diverse Treiber z.B für die EPSON LQ- und die NEC Px-Serie angeboten, die allerdings alle zusätzlich zum normalen Anschaffungspreis bezahlt werden müssen.

Verfügt man über den PostScript-Interpreter *UltraScript*, lassen sich entsprechende Druckdateien auch auf nicht-PostScript-fähigen Druckern ausgeben. Leider muß man für UltraScript die Dateien erst auf Atari-Format konvertieren, da es sich ja um ein Atari-Programm handelt. Wesentlich praktischer ist die Lösung über das Mac-Programm *TScript*, das für DM 269,- angeboten wird. Hier spart man die Konvertierung und erreicht aus jedem Mac-Programm Ausdrucke von hoher Qualität. Auch andere Bildschirm- und Druckunterstützungen wie z.B. *Adobe TypeManager* oder *TypeAlign* laufen auf dem Spectre einwandfrei.

Datenaustausch

Es wurde ja bereits die Möglichkeit des Datenaustausches zwischen Atari und Mac angesprochen. Dazu wird beim Spectre ein Programm namens *Transverter* mitgeliefert, mit dem man binäre und ASCII-

Dateien übertragen kann. Natürlich kann man auch den unbequemen Weg der seriellen Schnittstelle wählen, aber der ist langsamer und unpraktisch. Apple liefert auf seinen Systemdisketten z.B. ein Programm (*Datei konvertieren*) mit, mit dem man MS-DOS-Disketten lesen und beschreiben kann. Es empfiehlt sich übrigens, die DOS-Disketten mit diesem Programm zu formatieren, da sie ansonsten manchmal nicht erkannt werden. Ferner bietet dieses Programm eine Zeichenkonversion an, da der Mac-Zeichensatz nicht dem eines PC bzw. Ataris entspricht (Umlaute etc.). Eine weitere Möglichkeit bietet ein Mac-Programm, das MS-DOS-Disketten direkt vom Finder aus lesen und beschreiben kann, aber keine Zeichenkonversion bietet. Dabei handelt es sich um den *DOS-Mounter* der Firma Dayna. Die letzte uns bekannte Möglichkeit ist das Programm *Access PC*, über das wir aber keine weiteren Informationen haben.

Zukunft

Auf der letzten CeBIT konnte man schon Teile der Zukunft des Spectre bewundern. Dort wurde zum einen eine Erweiterungsplatine namens *MegaTalk* für Mega STs (keine Mega STEs) gezeigt, mit der ein AppleTalk-Low-Cost-Netzwerk unter dem Spectre möglich wird. Dadurch kann man dann z.B. ein Netzwerk aus Macs und Ataris aufbauen und alle Vorteile (gemeinsamer Drucker, Datenaustausch etc.) nutzen. Außerdem verfügt das MegaTalk-Board noch über eine SCSI-Schnittstelle, mit der man weitere Peripherie anschließen kann.

Vielleicht gelingt es den Programmierern, sogar TTs und Mega STEs direkt zu integrieren. Diese verfügen ja über eine LAN-Schnittstelle und den gleichen seriellen Baustein, über den beim Mac AppleTalk läuft. Allerdings ist das wirklich Zukunftsmusik, da auf der Atari-Seite keinerlei Software-Unterstützung für solch ein Vorhaben vorhanden ist.

Ebenfalls in Kürze wird ein 68030-Board mit 33 MHz, eigenem TOS 1.6 und zusätzlichem RAM-Speicher, der dann auch für die Atari-Seite zur Verfügung steht, zu haben sein. Die Herstellerfirma Gadgets by Small wirbt schon jetzt damit, daß der Emulator schneller als sein direktes Vorbild, der Mac Plus, sei. Mit dem Beschleuniger-Board wird dieser Geschwindigkeitsvorteil noch wesentlich ausgebaut.

Noch im Juni soll übrigens die Version 3.1 des Spectre auf den Markt kommen, in der dann u.a auch das TT-RAM unterstützt werden soll. Nichtsdestotrotz kann man sich bei Apple beruhigt zurücklehnen, denn zum einen verkaufen sich ihre Low-Cost-Rechner recht gut, zum anderen bleibt der Vorteil, daß es keine Versorgungsschwierigkeiten mit ROMs gibt.

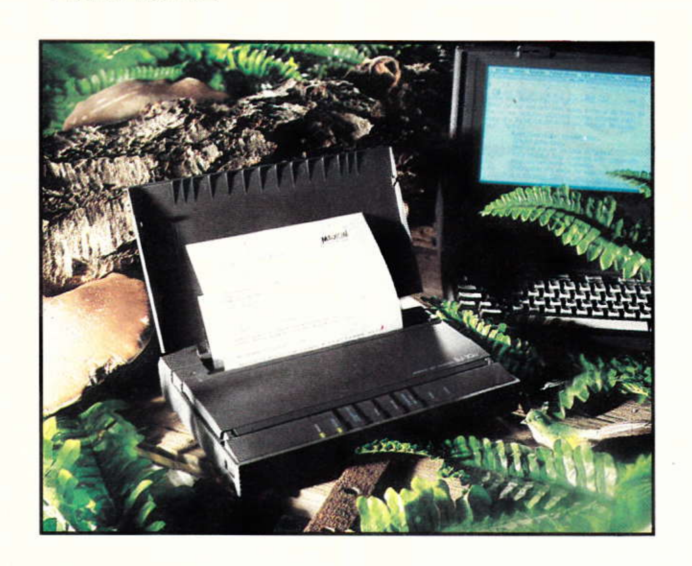
Der Spectre bietet für Atari-Besitzer mit seinem Anschaffungspreis von DM 570,- (ohne ROMs) sicherlich eine gute Möglichkeit, die Mac-Welt auf seinen Schreibtisch zu holen. Er arbeitet sehr betriebssicher und verfügt über eine hohe Kompatibilität. Professionelle Anwender werden aber garantiert das Original vorziehen, da hier die Probleme des GCR wegfallen und man auch mit Blick auf die Zukunft auf der sicheren Seite ist. Nicht zu vergessen, daß die Entwicklung des Mac mittlerweile schon um Längen weiter ist. Das Flaggschiff, der Mac IIfx, hat einen 68030 und ist mit 40 MHz getaktet, ein 68040-Rechner ist angekündigt. Natürlich hat das bei Apple alles seinen Preis, aber Geld spielt halt im professionellen Einsatz selten eine Rolle.

HE

Bezugsadresse:
HG Computersysteme
Giselastr. 9
W-5100 Aachen
Tel. (0241)603252

Mitnahme-artikel

Tintenstrahler Canon BJ-10e



Wessen Wahl auf einen tragbaren Computer fällt, der hat es entgegen aller Annahmen gar nicht leicht. Die Auswahl passender Peripheriegeräte ist nämlich noch längst nicht so groß, wie das Angebot der Laptops es erwarten läßt.

Was für Mäuse, Modems und Monitore gilt, trifft ebenfalls für das beliebteste und notwendigste Nebenaggregat, den Drucker, zu. Die Not, über keine tragbaren Geräte zu verfügen, hat für kurze Zeit sogar zu einer Renaissance kleiner 9-Nadler geführt.

In diese Marktlücke stößt Canon mit einem völlig neu konzipierten Winzling, und es sieht ganz so aus, als werde der knapp 2 Kilogramm schwere BJ-10e zum echten Renner. Wir haben uns den Tintenstrahler ebenfalls genauer angesehen und an den momentan einzigen behenkelten ST, den Stacy, angeschlossen.

Blasenbildung

Das Druckverfahren, das unser Fliegengewicht benutzt, wird von Canon 'Bubble-Jet Drucksystem' genannt. Ein Thermoelement erwärmt in einem Röhrchen Tinte, die sich daraufhin ausdehnt und am Rohrende, der Düse, eine Blase (Bubble) bildet. Die Blase trennt sich ab und tritt den Flug in Richtung Papier an, wo sie als Tropfen ankommt.

Canon kann auf langjährige Erfahrung mit dieser Technik zurückblicken; bereits in [1] hatten wir mit einem Drucker die Ehre, der auf diese Weise Tinte verspritzt. Doch ist das System bei weitem nicht unverändert geblieben. Der Art, wie Hewlett-Packard beim DeskJet [2] das Problem eintrocknender Tinte und verschlissener Düsen gelöst hat, schließt sich auch Canon an. Der BJ-10e besitzt eine Tintenpatrone, in der die 64 Düsen bereits eingebaut sind. Sie werden mit der kompletten Kartusche ausgetauscht.

Im großen und ganzen ist der kleine Canon ein 'normaler' Drucker. Das Papier wird klassisch um die Druckwalze gelegt, die Einzelblattzuführung erfolgt von oben. Die Verarbeitung perforierten Materials ist allerdings nicht möglich - für Traktoren ist im Winzling kein Platz. Eine parallelle Schnittstelle ist eingebaut, das Netzteil jedoch ist extern - Tribut an die Gewichtsreduzierung. Dafür aber findet sich an der Rückseite ein Fach für den als Sonderzubehör erhältlichen Akku. Der soll nach einer Ladezeit von 10 Stunden den Drucker für 40 Minuten mit Energie versorgen. Wenn man bedenkt, daß so mancher Laptop auch nur zwei Stunden netzunabhängigen Betriebes erlaubt, scheint das völlig ausreichend.

Schreihals

Ob es zur Energieeinsparung beiträgt oder schlicht nur preiswerter ist, sei dahingestellt. Mich jedenfalls nervt das heisere Gepiepse, mit dem der BJ-10e seine Einstellungen kundtut. Mit einer der gut geformten Tasten auf dem Panel kann man - je nach Betriebsart - Schriftstil und -breite einstellen. Weil dafür auf Lämpchen verzichtet wurde, besteht die Kontrolle der Einstellung nur in einem Piep. Der Blick ins Handbuch („Wie oft drücken war nun Fettdruck?") bleibt einem nicht erspart. Hier könnte ein Aufkleber auf der Innenseite der Klappe für Klarheit sorgen.

Die Tastatur ist gut geraten, auffällig sind die Knöpfe zum Vor- und Zurückfahren des Papiers. Dem BJ-10e fehlt aus Platzgründen der Walzendrehknopf, da kann das Papier auf diesem Weg von Hand bewegt werden. Unter der Klappe finden wir auch zehn DIP-Schalter, gegen die an dieser Stelle nichts einzuwenden ist.

Die Bedienung des kleinen Canon ist völlig unproblematisch und simpel. Das Papier zieht er halbautomatisch ein. Das heißt: Papier auf die Klappe legen, die hochgestellt als Rutsche dient, Knopf drücken, und schon kann's los gehen. Der Druckbeginn ist aufgrund der sehr kompakten Bauweise erfrischend weit oben auf dem Blatt, so daß der BJ-10e mit 68 Zeilen ein DIN A4-Blatt sehr gut ausnutzt.

Akrobatik

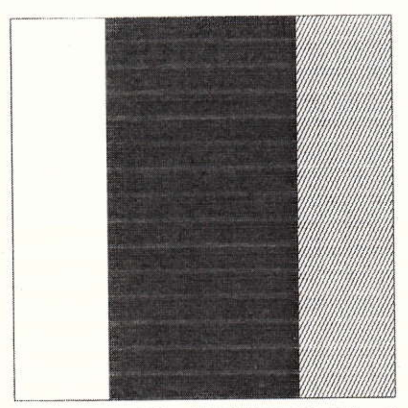
Ein besonderer Gimmick, der gar nicht so lächerlich ist, wie er scheint, ist der Ständer an der Rückseite des kleinen Druckers. Er wird gedreht, schon steht das Gerät auf dem Rücken. Von der Unterseite, die im Bild rechts ist, wird durch einen Schlitz stärkeres Papier zugeführt. Canons Ingenieure haben hier die Not zur Tugend gemacht. Die kleine und damit schwache Mechanik des Druckers scheitert im normalen Betrieb an Umschlägen. Durch den Schlitz aber erhält der BJ-10e nahezu Flachbettdruckerqualitäten. Auch verstärkte Umschläge zieht er ohne Klagen ein. An der Rückseite wird auch der als Option erhältliche Einzelblatteinzug angesetzt.

Blue Mother's treuer Gesell'

Eher bescheiden zeigt sich der Zwerg auf

```
Canon BJ-10e
Hoch auf dem gelben Wagen sitz ich bei Schwager vorn
```

Schriftprobe



Grafikprobe: 360 x 180 DPI Unidirektional gedruckt





seiner Software-Seite. Ich rückte ihm mit dem Druckertestprogramm (PD 420/421) zu Leibe, doch häßlich ward das Ergebnis. Der Grund: Die einzige Emulation, die in seinem Innern schlummert, verhilft ihm dazu, die Befehle eines IBM-XL24-Druckers zu verstehen. Welche Verschwendung! Der Drucker versteht auf irgendeine Weise auch Canon-eigene Grafikbefehle für hohe Auflösungen, nur welche Software beherrscht die? (Leider läßt sich das Handbuch über diese Kommandos nicht weiter aus.) Doch in der IBM-Emulation ist normale 360(DPI waagerecht) mal 180(DPI senkrecht)-Grafik die Grenze des Machbaren. Ein Vorschub um ein 360tel Zoll fehlt nämlich.

Mit gängiger ST-Software gibt's also zwei Reibungspunkte. Zum einen im normalen Textbetrieb. Dort bedarf es Treiber für IBM-Drucker, die entsprechende Kommandos zur Umschaltung der Schriftbreite und des Stils verwenden. Gängig am ST sind Treiber für Epson-kompatible Geräte, doch für viele Textverarbeitungen gibt es auch IBM-Treiber, zumal die Anpassung nicht sehr problematisch ist.

Zum anderen ist da die Grafik. Zunächst muß man sich auf 360*180 Punkte pro Zoll beschränken, siehe oben. Weitere Probleme gibt's dann nicht mehr, wenn das Text- oder Grafikprogramm die richtigen Grafikbefehle benutzt ('ESC * ...') und die Positionierung mit 'ESC $ ...' vermeidet sowie 'ESC 3' zum Setzen des Zeilenabstandes benutzt. Bei Signum! läßt sich letzteres jedoch nicht abschalten, so daß der Druck mißlingt. Sind diese Voraussetzungen allerdings erfüllt (Tempus Word, Script, Arabesque) gelingt die grafische Datenausgabe problemlos.

Wenn der geneigte Leser einen Blick in die Tabelle wagt, wird er gewahr, daß es sich beim BJ-10e um ein flinkes Kerlchen handelt. Er bricht zwar keine Rekorde, aber im guten Mittelfeld preiswerter 24-Nadler hält er mit. Dabei ist der Tintendruck längst nicht so unproblematisch, wie man denkt. Denn das schwarze Naß muß Zeit zum Trocknen bekommen. Daher fällt auch beim Canon der relativ langsame Zeilenvorschub auf. Zudem bedarf es ausgeklügelter Maßnahmen, um die Düsen während des Druckens vorm Eintrocknen zu bewahren. Dazu fährt der Canon seinen Kopf nach wenigen Zeilen in eine Ruheposition, wo eine Verschließ- und Auffangvorrichtung auf ihn wartet. Dort pustet er mit allen Düsen hinein, um sie in Gang zu halten, und weiter geht's. Bei jeder kleineren Druckpause bewegt er sich ebenfalls dorthin, um die Düsen zu verschließen.

Nervenschoner

Geschwindigkeit ist natürlich nicht alles, der Canon besticht zudem durch seine geringe Geräuschkulisse. Ein sanftes Klacken bei jeder Kopfbewegung ist alles, was er von sich gibt. Und das stammt vom Spindelantrieb, den Canons Ingenieure dem Winzling zum Kopfantrieb verpaßt haben. Recht ungewöhnlich, ein solcher Aufbau. Doch ist er vermutlich unempfindlich gegenüber rüder Behandlung unterwegs. Apropos mitnehmen: Der Druckkopf wird bei jedem Ausschalten des Gerätes arretiert, so daß auch er gegen Stöße gefeit ist.

Weit weniger schonend geht der BJ-10e mit dem Geldbeutel seines Herrn um. Die Patronen mit der Tinte kosten per Stück 59 DM und sind laut Canon gut für 700.000

Drucker	Text Draft-Endlos	Text LQ-Endlos	Text LQ-Einzel	Text Brief	Grafik Brief 180	Grafik Brief 360	Grafik ST-Hardcopy	Grafik S/W-Bild	Grafik Farbbild	Kopfbe-schleun.	Ko-pien
Canon BJ-10e	-	-	00:18 / 13:41	00:42	01:06	-	00:17 / 00:21	-	-	0%	-
Citizen Swift 24	03:40 / 04:44	08:58 / 11:39	-	00:41	00:54	01:46	00:23 / 00:25	-	-	-	1+1
Fujitsu DL 1100	01:19 / 05:11	03:08 / 12:12	03:57 / 15:35	00:46	00:44	01:23	00:15 / 00:44	05:33	88%	1+3	
Seikosha SL-92	00:19 / 04:54	00:19 / 10:11	00:19 / 12:28	00:37	01:12	02:15	00:18 / 00:19	-	-	81%	1+3
STAR LC24-200 COLOUR	00:38 / 04:50	00:43 / 11:37	-	00:42	00:47	01:30	00:21 / 00:26	02:48	06:20	0%	1+4

Zeichen. Das sind rund 550 DIN-Briefe und damit Verbrauchskosten pro Brief von ca. 11 Pfennigen. Damit liegt er höher als viele Laserdrucker, zumal darin nicht die Abnutzung des Gerätes enthalten ist (ebensowenig wie das Papier).

Alleskönner

Rundherum ist der Canon BJ-10e eine sehr erfreuliche Maschine. Den Japanern ist es gelungen, eine ganze Reihe von pfiffigen Lösungen in einem Gerät unterzubringen, das dabei nicht überteuert ist. Die vielseitige Papierverarbeitung, die relativ hohe Geschwindigkeit, die Geräuscharmut erfüllen meine Ansprüche an ein ergonomisches Gerät. Gleichzeitig ist es klein, leicht, sogar netzunabhängig und außerdem in angenehmem Grau gehalten.

Nicht nur neben dem Laptop macht er sich auch auf dem Schreibtisch gut. Der Canon ist das richtige Gerät für denjenigen, der nicht viel zu drucken hat, sich dann nicht vom Krach des Nadlers nerven lassen will. Zwischendurch kann man ihn sogar in der Schublade verschwinden lassen. Die Qualität, die er zu Papier bringt, können Sie in der Grafik- und Schriftprobe besichtigen. An ihr ist nichts auszusetzen, sie ist vor allem gleichmäßiger als bei einem Nadeldrucker.

Kritikpunkte sind die hohen Tintenkosten und die mangelnde Software-Ausstattung. 37 kB Pufferspeicher sind zwar ein Trostpflaster, doch eine Epson-Emulation würde ihm gut zu Gesicht stehen. Wer ihn in sein Herz geschlossen hat und noch ein wenig warten will, für den wird es noch in diesem Sommer ein identisches Gerät von Brother geben, das vor allem durch bessere innere Werte glänzt (siehe auch [3]). Auch die Apfelmännchen aus Palo Alto fanden Gefallen an Canon's Kleinstem: Im neuen Apple StyleWriter, dem Low-Cost-Gerät zum Mac Classic, befindet sich ebenfalls das Druckwerk des BJ-10e. Der zukünftige Besitzer kann davon nur profitieren. Denn so sind Angebot und Wettbewerb für das Verbrauchsmaterial gewährleistet.

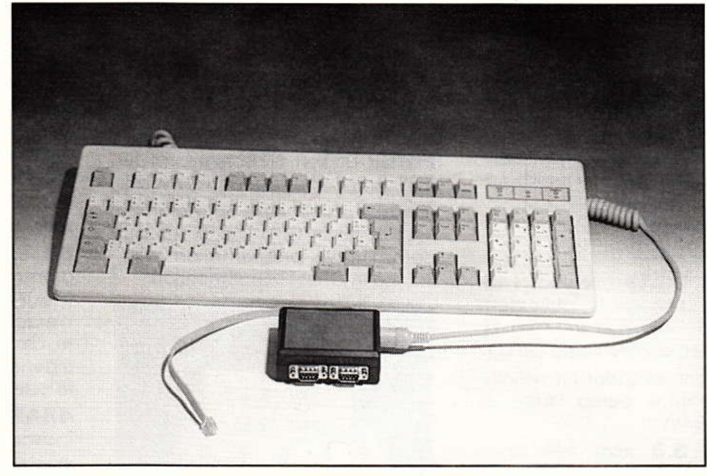
IB

Canon BJ-10e

Grundgerät:	998 DM
Akku:	128 DM
Tintenpatrone für 700.000 Zeichen:	59 DM
Garantie:	6 Monate

[1] Canon BJ-130, ST-Computer 7,8/89, S.62 ff.
[2] Lärmschutzmaßnahme - HP DeskJet PLUS, ST-Computer 4/90, S.54 ff.
[3] Beeindruckend - Neue Drucker auf der CeBIT 91, ST-Computer 6/91, S. 60 ff.

Der ST/TT tippt fremd!



Das zweifellos Wichtigste an einem guten Computer sind zum einen die Bildschirmqualität und zum anderen eine gute Tastatur. Der Atari ST erfüllt die erste Bedingung durch seinen monochromen Monitor mit dem gestochen scharfen Bild mit Bravour. Leider läßt seine Tastatur (zumindest bei den 260/520/1040 STs) doch sehr zu wünschen übrig.

Schwammiger, unpräziser Anschlag, kein Druckpunkt nie weiß der Anwender genau, ob sein Zeichen auch wirklich vom Computer übernommen wurde. Auch die Tastatur bei den Mega ST-Computern ist noch nicht der Weisheit letzter Schluß, weil die Tastenkappen unüblich groß ausgefallen sind. Tippfehler sind die üblen Folgen dieser Spar- und Design-Maßnahme von Atari. Einige findige Firmen haben sich mit Notlösungen, wie z.B. neuen, anders geformten Tastenkappen beholfen, die einfach gegen die alten ausgetauscht werden. Dies brachte schon eine gewisse Verbesserung, aber von einer echten professionellen Tastatur, wie man sie von PCs gewöhnt ist, kann dabei immer noch nicht die Rede sein. Abhilfe kann hier nur eine Hardware-Lösung schaffen, die es erlaubt, normale PC-Tastaturen an den ST anzuschließen. Auch auf diesem Gebiet gibt es mittlerweile einige Anbieter. Meistens wird dabei ein Interface an den ROM-Port oder gar an den MIDI-Port angeschlossen. Dies blokkiert nicht nur wichtige Schnittstellen, es ist auch spezielle Software zum Betrieb der PC-Tastatur notwendig und- wie sich jeder denken kann- geht das nicht immer ohne Probleme ab. Selbststartende Spiele oder Anwenderprogramme, welche direkt den Tastaturprozessor des Atari ansprechen (auch das gibt es!), können leider nicht mit einer solchen Tastatur zusammenarbeiten. Auch Hardware-Emulatoren, wie beispielsweise der Spectre 3.0 oder AT-Speed, funktionieren dabei nicht, da sie eigene Routinen zum Abfragen der Tastatur benutzen. Hier gibt es nur einen Weg: es muß ein Interface her, das an die Tastaturschnittstelle angeschlossen wird und den originalen Tastaturprozessor des ST beinhaltet. Nur dadurch kann eine 100%ige Kompatibilität zur normalen ST-Tastatur gewährleistet werden. Einen Vertreter dieser Sparte haben wir zum Test zur Verfügung gestellt bekommen. Es ist das PC-Tastatur-Interface der Firma HG-Computersysteme. Ein Blick auf das ca. 5 x 7 x 3 cm kleine, schwarze Kunststoffkästchen läßt nicht vermuten, was alles in ihm steckt. Ein fest angeschlossenes Kabel führt heraus und Buchsen für Maus, Joystick und eine normale 5 polige DIN-Buchse zum Anschluß beliebiger PC-Tastaturen sind zu erkennen. Das Kabel endet in einem (Atari üblichen) amerikanischen Telefonstecker. Besitzer eines Mega ST, Mega STE oder TT können also gleich, ohne ihren Computer zu öffnen, das Interface anstelle der ST-Tastatur anschließen. Lediglich den Tastaturprozessor (6301) muß man aus dem original Atari-Keyboard entfernen (hierzu die Atari-Tastatur öffnen und den Chip vorsichtig aus dem Sockel hebeln) und in den dafür vorhandenen Sockel des Interfaces einsetzen. Wichtig dabei ist, daß der Prozessor korrekt in den Sockel gedrückt wird, (Markierung auf dem Chip seitengleich mit der Nase des Sockels). Bei verkehrtem Einbau ist es ziemlich sicher, daß der Chip sich binnen kürzester Zeit ins „Nirwana" verabschiedet. Diesen Umbau kann man sich allerdings ersparen, wenn man einen Tastaturprozessor (gegen Aufpreis) bei HG-Computersysteme mitbestellt. Dieser ist dann gleich in dem Interface eingebaut. Bei Computern der Serie 260/520/1040 ST bzw. 1040 STE ist der Anschluß ungleich komplizierter. Zunächst muß man den Computer öffnen und den Stecker der eingebauten Tastatur von der Hauptplatine abziehen. Auch hier braucht man den Tastaturchip nur umzustecken, wenn man keinen extra mitbestellt hat. Zusätzlich muß aber auch das Kabel am Interface ausgetauscht werden. HG-Computersysteme liefert zu diesem Zweck ein Spezialkabel mit. Dieses verbindet direkt die ST-Hauptplatine mit dem Interface. Leider gibt die Anleitung keinen Hinweis, wie das Kabel aus dem ST-Gehäuse herausgeführt werden soll. Man kommt nicht darum herum, in die Werkzeugkiste zu greifen und sich selber einen „Ausgang" aus dem ST-Gehäuse zu feilen.

Doch nun zum Praxistest. Bei uns verrichtete das Interface auf Anhieb seinen Dienst. Die ebenfalls bei HG-Computersysteme erhältliche Cherry-Tastatur zählt ohne Zweifel mit zu den besten Tastaturen auf dem Markt. Es lassen sich aber auch viele andere Keyboards verwenden, einzige Bedingung: Sie müssen zwischen XT und AT umschaltbar sein. Benutzt wird die Stellung „XT" (bei manchen Tastaturen auch „PC" genannt). Die Belegung solcher Tastaturen entspricht nicht ganz dem Atari-Standard. Es gibt z.B. keine UNDO- und HELP-Tasten, dafür aber zwei weitere Funktionstasten (F11 & F12). Es lag also nahe diese anstelle der HELP- und UNDO-Tasten zu verwenden. Auch die Sonderzeichen sind bei dem Atari-Keyboard teilweise anders plaziert. Hier machten die Entwickler des Tastatur-Interfaces „Nägel mit Köpfen" und ließen sowohl die original ST-Tastenkombinationen (z.B. SHIFT-ALT-"Ö" = geschweifte Klammer) als auch die auf der PC-Tastatur aufgedruckten Zeichen (ALT-"7" = geschweifte Klammer) zu. Falls es aber ein Programm geben sollte, welches gerade diese Tastenkombinationen für sich benutzt, lassen sich die emulierten Sonderzeichen auch abschalten, so daß sich die PC-Tastatur 100%ig wie eine ST-Tastatur verhält. Mit diesen Möglichkeiten ist man wirklich für alle noch so exotischen Programme gerüstet. Uns ist während der Testphase auch kein Programm

untergekommen, bei dem die Tastatur ihren Dienst versagt hätte. Zusätzlich zu der normalen und der speziellen Sondertastenbelegung ist auch noch ein neuer Cursor-Tasten-Modus integriert worden. Schaltet der Anwender die „NUM-LOCK“-Funktion ein, leuchtet die dazugehörige Leuchtdiode in der Tastatur auf. Mit den Cursor-Tasten wird nun der Mauszeiger anstelle eines Cursors in groben Schritten bewegt. Hier wird also quasi eine Art „ALT-LOCK“-Funktion emuliert. Den Tasten des Ziffernblock kommt dann auch eine neue Bedeutung zu. Mit den Ziffern „4“,“8",“6“,“2" kann der Mauszeiger pixelweise bewegt werden, „+“ und „ENTER“ simulieren dann die rechte bzw. linke Maustaste. Besonders bei Zeichen- und CAD-Programmen bringt das erhebliche Vorteile. Pixelgenaues Positionieren mit der Maus ist immer ein anstrengendes Unterfangen für Augen und Hände. Per Tastatur kann dies z.T. erheblich einfacher und schneller gehen.

Neben den Sonderzeichen wertet das Interface auch einige Tastenkombinationen als Sonderfunktionen für sich aus. So läßt sich z.B. ein Hardware-Maus-Speeder aktivieren oder ein Reset nur für das Interface auslösen. Letzteres kann des öfteren zum „Retter in der Not“ werden, wenn mal der Tastaturprozessor (6301) „abstürzt“. In der Regel hilft dann nur noch der „Affen-Griff“ zum Reset-Taster des Computers, aber wichtige Daten gehen unter Umständen verloren. Hier kann der Tastaturprozessor durch die Kombination: rechte ALT-Taste; rechte CTRL-Taste; „DELETE“, meist wiederbelebt werden, ohne daß der Computer selbst einen Reset durchführt. Im Test traten mit dieser Funktion allerdings ein paar kleine Probleme auf. Die Tastatur schien nach einem solchen Reset nicht mehr richtig zu funktionieren. Wirre Zeichen wurden zum Computer gesandt, und der „NUM-Lock“-Modus blieb permanent aktiv. Durch einen Kalt- bzw. Warmstart des Computers ließ sich dies aber beheben. Ein einziger Wunsch bleibt offen. Es wäre sicherlich das Tüpfelchen auf dem I, wenn sich die Tastenbelegung nach eigenen Vorstellungen anpassen ließe. So ist man leider auf die vorgegebene Belegung beschränkt, aber die o.a. Lösung würde sich sicherlich unverhältnismäßig hoch auf die Kosten des (so schon recht teuren) Interfaces auswirken.

Fazit: „100 Prozent kompatibel zur Atari-Tastatur!“, versprach die Werbung. Dies können wir nach diesem Test auch bestätigen. Weder exotische Programme oder Spiele, noch PC- oder Mac-Emulatoren verweigern die Zusammenarbeit mit dem HG-Interface und der Cherry-Tastatur. Die durchdachte Lösung bezüglich der Sonderzeichen, die nützliche „Cursor-Maus-Emulation“ und die hohe Betriebssicherheit führen dazu, daß wir das Gerät durchaus empfehlenswert finden. Vielleicht wird nicht jeder Atari ST-Anwender eine Profi-Tastatur benötigen, aber für „Vielschreiber“ und „rasende Reporter“, die täglich Texte kilobyteweise in die Tasten hämmern, ist das Interface von HG-Computersysteme eine praktikable Lösung, zudem auch noch kompatibel zu allen Anwendungen und Spielen. Lediglich der Preis läßt den Normalanwender etwas zurückschrecken. 189,- DM für das Interface (mit Handbuch) plus ca. 200,- DM für eine professionelle Tastatur (Cherry) schlagen doch kräftig zu Buche. Für Anwender, die ihren ST lieber nicht aufschrauben wollen, fallen zusätzlich 50,- DM für den Tastaturprozessor an. Damit liegt der Gesamtpreis für eine Komplettlösung bei über 400,- DM. Klar, daß hier der professionelle Markt angesprochen wird. Hierfür ist aber ein solches Instrument beinahe unverzichtbar.

CM

Bezugsadresse:
HG Computersysteme
Giselastr. 9
W-5100 Aachen
Tel. (0241)603252



Die digitale Justitia

Sound-Sampling

Die Geschichte der Kunst ist auch eine Geschichte der Plagiate. Angefangen von Variationen bekannter Klassiker (z.B. Brahms über ein Thema von Joseph Haydn {op.56}), zieht sich der rote Faden des Ideenklaus durch die Musikbranche bis zur heutigen Zeit.

Die Technik macht jedoch alles einfacher. Mußte man sich bei einem Plagiat früher noch Gedanken darüber machen, wie der Urheber die Noten gesetzt hat, so erledigt dies heutzutage der Computer selbst. Das Zauberwort hierzu heißt: SAMPLING.

Problemstellung

Gerade die Computer der Atari ST-Serie sind in der Musikbranche „die" Computer schlechthin, und die Software-Industrie überschlägt sich mit neuen Produkten zur Soundgestaltung im Wege des Samplings.

Sampling ist die Möglichkeit Geräusche und Töne in digitale Zeichen umzuwandeln, die vom Computer gelesen und auch verändert werden können. Diese Tonfolgen lassen sich dann anschließend auch wieder hörbar machen. Ein Klangunterschied zum Original ist dabei praktisch nicht zu erkennen. Gleichzeitig können diese Klänge vom Computer gespeichert und damit jederzeit reproduziert und auch über das Speichermedium jedem zugänglich gemacht werden. Der weitere Vorteil liegt darin, kleinste Geräuschsequenzen herauszufiltern und für eigene Zwecke zu nutzen. Die dadurch eröffneten Möglichkeiten wurden in einer Musikzeitschrift kürzlich drastisch beschrieben:

„Da könnte sich Miles Davis auf einer Roland-Kaiser-Scheibe ein Solo spielen hören oder die fette Cozy-Powell-Baßdrum auf einer Nummer dahindampfen, von der Herr Powell nicht mal weiß, daß sie existiert"[1].

Gerade in der Rap- und Hip-Hop-Scene, die sich häufig aus Musikzusammenmischungen auszeichnet, werden ganze Musikstücke aus Teilen fremder Melodien zusammengewerkelt.

Derzeit werden die ersten Klagen eingeleitet - Gerichtsentscheidungen liegen jedoch bislang noch nicht vor. Auch in der rechtlichen Literatur hält man sich noch bedeckt, weil die Frage, ob die Verwendung von Sampling-Klängen rechtlich zulässig ist oder nicht, bislang völlig ungeklärt ist.

Urheberrechtliche Probleme

Das Sampling von Musikstücken kann grundsätzlich verschiedene Gründe haben. Es können dadurch Musikteile parodiert werden oder als Toncollage dienen, oder schließlich dazu benutzt werden, um Musikstücke, -teile oder Klangfolgen zum Zweck des „Klang-Klaus" zu kopieren. Diese verschiedenen Ansatzpunkte bedürfen auch der unterschiedlichen urheberrechtlichen Regelung.

Ausgangspunkt der Problematik ist § 24 Absatz 2 Urhebergesetz (UrhG). Nach dieser Vorschrift darf eine Melodie aus einem anderen Werk der Musik nicht entnommen und einem neuen Werk zugrundegelegt werden.

Subsumiert man alle oben aufgeführten Gründe des Samplings unter diese Norm, so kommt der unbedarfte Leser zwangsläufig zu dem Ergebnis, daß das Sampling grundsätzlich verboten ist. Dieses vorläufige Ergebnis bedarf jedoch einiger Einschränkungen.

So hat das Oberlandesgericht Dresden [2] in einer früheren Entscheidung die Überlegung aufgemacht, daß der unflexible Melodienschutz bei einer Persiflage nur dann anzuwenden sei, wenn das neue Werk objektiv geeignet sei, dem Originalwerk Konkurrenz zu machen und seinen Absatz zu beeinträchtigen. Daraus läßt sich schließen, daß das kopierte Musikwerk nur dann urheberrechtlich unzulässig ist, wenn es den gleichen Hörerkreis wie das Originalwerk hat, zeitlich parallel zu diesem auf den Markt gebracht wird oder ihm in sonstiger Weise erhebliche wirtschaftliche Konkurrenz macht.

Bezüglich des Samplings als Toncollage ist anzumerken, daß hier grundsätzlich der Musikschutz des §24 II UrhG greift. Melodiecollagen sind daher grundsätzlich keiner freien Benutzung zugänglich. Voraussetzung ist jedoch immer, daß es sich bei der Melodie um eine persönliche geistige Schöpfung handelt [3]. Dies ist aber dann fraglich, wenn nur kleine Musikteile (sog. Licks) verwertet oder als Vorlage benutzt werden. Aus diesem Grund kann ein Musikstück durchaus aus über 100 Einzelteilen zusammengestückelt sein. Bei solchen Einzelfällen erscheint dann eine Schutzwürdigkeit des originären Rechtsinhabers fragwürdig und eine gerichtliche Ahndung abwegig.

Der komplizierteste Fall ist jedoch das Sampling im Bereich des „Klang-Klaus". Hierbei werden einzelne Instrumentalbereiche (z.B. Schlagzeugfiguren, Baßläufe, Keyboardeinstellungen oder auch nur Stimmen oder Stimmteile) mit Sampling-Geräten kopiert und über Sound-Datenbanken gewerblich weiterveräußert. Ei-

nigkeit besteht in der Literatur noch nicht einmal darüber, ob diesen Musikteilen überhaupt urheberrechtlicher Schutz zukommen kann, weil es sich bei diesen Mitschnitten nicht um Melodien iSd. §24 II UrhG, sondern nur um Klänge handelt. Die Tendenz läuft jedoch zum Urheberrechtsschutz von Keyboard-Einstellungen und Stimmen, da diese im Gegensatz zu Schlagzeug und Baßläufen Klangfarbe besitzen und nicht reine Rhythmuselemente sind [4].

Bedenken bezüglich dieser Auffassung ergeben sich jedoch aus der Tatsache, daß gerade die Kunst der Popmusik weniger in der Melodiegebung als in der Rhythmisierung und der Ausnutzung besonderer Klangeffekte liegt. So erweisen sich bestimmte Sounds häufig als Erkennungsmerkmal bestimmter Künstler, die zeit- und kostenintensiv durch Studioaufnahmen produziert wurden. Somit besteht die Gefahr, daß die Popmusik ihrer kreativsten Teile beraubt werden kann. Im Ergebnis ist zunächst jedenfalls festzuhalten, daß ein Urheberrechtsschutz nur unzulänglich besteht.

Wettbewerbsrechtliche Probleme

Auch in wettbewerbsrechtlicher Hinsicht bestehen Bedenken. Ein Wettbewerbsverstoß liegt nämlich nach §1 des Gesetzes gegen den unlauteren Wettbewerb (UWG) dann vor, wenn im geschäftlichen Verkehr zu Zwecken des Wettbewerbs Handlungen vorgenommen werden, die gegen die guten Sitten verstoßen.

Das Merkmal „im geschäftlichen Verkehr“ ist im Fall des Samplings von Musikstücken oder -teilen unproblematisch zu bejahen. Das Merkmal „zu Zwecken des Wettbewerbs“ führt dagegen schon dann zu Problemen, wenn die nun digitalisierten Klänge von Künstlern herrühren, die in der Musikscene schon als veraltet gelten, weil sie auch nicht mehr am Musikwettbewerb teilnehmen. Soweit die Sampling-Produkte den gleichen Markt wie ihre Vorbilder ansprechen und den eigenen Absatz zu Lasten des Konkurrenten fördern können [5], liegt der Wettbewerbszweck vor.

Die größten Schwierigkeiten bereitet jedoch das Merkmal der „Sittenwidrigkeit“. Zwar ist das Kopieren fremder Arbeitsergebnisse nach herrschender Meinung eine sittenwidrige Wettbewerbshandlung, da ein solches Schmarotzen den Mitbewerber um die verdienten Früchte seiner Arbeit bringt [6]. Das fremde Arbeitsergebnis durch das Sampling muß jedoch identisch oder nahezu identisch mit dem Original sein. Zwar entsteht nach dem Sampling eine identische Kopie des Originals. Dieses Arbeitsergebnis wird jedoch nicht unbedingt direkt in das neue Musikstück übernommen. Nur dann läge eine sittenwidrige Wettbewerbshandlung vor. Vielmehr werden aber nach Transformation und Modifizierung von kleinen Tönen oder Tonfolgen häufig neue Tonsequenzen geschaffen, die durch zusätzliche Programmierung der Entlastung des Musikers zugunsten komplizierter und origineller Sequenzen dienen [7]. Soweit somit Unterschiede in der tatsächlichen Verwertung des Originals nach dem Sampling-Vorgang bestehen, ist der Nachweis einer Wettbewerbsverletzung gering.

Auch wenn eine Kopie nachgewiesen wurde, steht der Unwertgehalt noch nicht fest, da nach herrschender Meinung eine Interessenabwägung erforderlich ist [8]. So kann trotz Vorliegens einer direkten Kopie das Interesse des Betroffenen zurückgestellt werden, wenn der „Sound“ des Musikers in Vergessenheit geraten ist und als Oldie vor sich hin modert. Der Leistungsschutz des §1 UWG garantiert ihm nämlich nur einen begrenzten Zeitraum zur eigenen Nutzung seiner Leistungen [9]. Die Dauer der Alleinnutzungsfrist kann allerdings nicht abstrakt festgelegt werden. Hier ist vielmehr auf den Einzelfall abzustellen, in welchem der Erfolg eines Musikstückes (sei es als aktueller Hit oder als Oldie) bezüglich des aktuellen Umsatzes Einfluß auf die Dauer der Nutzungsberechtigung ausübt.

Sampling für Programme

Gerade bei Spielen finden sich mitunter faszinierende Melodien, die einen durchaus bekannt vorkommen können. Die Programmierer machen sich hierbei nicht unbedingt die Mühe, sich zur Programmierarbeit eines Spiels auch noch Melodien auszudenken. Viel einfacher ist es, auf bekannte Musikstücke zurückzugreifen und zur entsprechenden Spielsituation die passende Melodie aus dem Fundus des Bekannten auszuwählen. Was liegt daher näher, als das Sampling bekannter Musikstücke?

Hierzu gilt jedoch grundsätzlich das oben Gesagte. Eine unterschiedliche rechtliche Beurteilung eines Musikers, der sich fremde Ideen zu eigen macht, und dem Programmierer, der das gleiche unternimmt, ist unangemessen. Daher liegt auch beim Sampling von Musikstücken zum Zweck der Programmiereinbindung eine Urheberrechtsverletzung vor. Die oben genannten Einschränkungen gelten selbstverständlich auch hier.

Allerdings ist zu bedenken, daß eine Wettbewerbsverletzung hier nicht vorliegen kann, weil der Programmierer nicht zu Zwecken des Wettbewerbs tätig wurde. Zwar geht er mit seinem Programm auf den Markt, um dieses mit größtmöglicher Gewinnspanne zu veräußern. Jedoch nimmt der Programmierer nicht am Musikwettbewerb teil. Aus diesem Grund scheidet ein Wettbewerbsverletzung aus. Ein Wettbewerbsverstoß liegt jedoch dann vor, wenn das durch Sampling kopierte Musikstück selbst aus einem anderen Programm stammt, oder wenn die Musikeinlage disassembliert und in das eigene Programm eingebaut wurde. In diesem Fall liegt der Verstoß nämlich auf der gleichen Wettbewerbsebene. Dann steht dem Betroffenen auch ein Unterlassungs- und Schadensersatzanspruch zu, wenn er selbst Urheber der Melodie ist und sie nicht selbst durch Sampling oder andere Methoden wettbewerbswidrig „geklaut“ hat.

Zusammenfassung

Die rechtliche Beurteilung des Samplings ist durch die Rechtsprechung noch nicht eindeutig geklärt. Vieles ist ungeklärt und strittig. Aufgrund dieser Unsicherheit gibt es zur Zeit kaum Künstler, die gegen das Sampling anderer vorgehen, da das Kostenrisiko noch unabwägbar ist. Ein Ausweg bietet sich jedoch dann, wenn ein Musterfall beispielsweise durch die GEMA durchgezogen wird, anhand dessen sich der gordische Knoten der Rechtsunsicherheit auflösen wird. Man darf gespannt sein.

Rechtsanwalt Christoph Kluss
Kurhessenstr. 31
6000 Frankfurt/Main 50
Tel. (069)531092

[1] M.Brem, Sampling.
Die Revolution aus dem Mikro-Chip.
In ME/Sounds 3/1987, 67-71 (69)
[2] OLG Dresden in GRUR 1909/332 (336 ff.)
Selbstverständlich bezog sich diese Entscheidung nicht auf das Soundsampling, sondern nur auf das „normale Kopieren“ einer musikalischen Idee.
[3] vgl. Schricker/Gerstenberg, Urheberrechtskommentar, München 1990 §24 Rz.23
[4] ebd.
[5] BGHZ 19/392 (393)
[6] Baumbach/Hefermehl, UWG-Kommentar, §1 Rz.444 f.
[7] Brem aaO. S.68
[8] Baumbach/Hefermehl, aaO., §1 UWG, Rz.449
[9] Bundesgerichtshof in BGHZ 51/41 (48)

Programmier Praxis

INHALT

TURBO C BLUES

Friedel van Mengen

TURBO C IST JA EIN ANERKANNT GUTER C-COMPILER FÜR DEN ATARI. EINEN GUTEN TEIL SEINER POPULARITÄT HAT ER AUCH SEINER INTEGRIERTEN ENTWICKLUNGSUMGEBUNG ZU VERDANKEN. DIESE ARBEITET UNTER 'NORMALEN' UMSTÄNDEN AUCH PRÄCHTIG. DOCH LEIDER HABEN IMMER WENIGER COMPUTERBENUTZER EINEN 'NORMALEN' RECHNER. VIELE HABEN IHREN RECHNER HIER UND DA EIN WENIG ERWEITERT UND MÖCHTEN DIESE ERWEITERUNGEN MIT ALLEN PROGRAMMEN NUTZEN KÖNNEN.

Vor einigen (vielen) Monaten wurde im ST-Magazin [1] ein Beitrag veröffentlicht, der versprach, dem Atari-Monitor SM124 hinsichtlich der Bildschirmauflösung Beine zu machen. Diese Hyperscreen-Erweiterung funktionierte auch bestens, und mit dem abgetippten Listing gelang es auch schließlich, (etwas) Großbildschirm-Feeling aufkommen zu lassen.

Doch leider arbeitet Turbo C mit Hyperscreen NICHT. Der eingebaute Editor arbeitet ohne Schwierigkeiten (wenn man davon absieht, daß er viel zu langsam ist), aber beim Umschalten auf den User-Screen mittels ESC erscheinen allerlei unsinnige Muster auf dem Bildschirm; nach nochmaligem Umschalten muß man, um einen Absturz des Programms zu vermeiden, sofort QUIT anwählen (keine gute Alternative...) Dies ist umso verwunderlicher, da die Shell auch auf Großbildschirmen läuft. Dort wird beim Umschalten auf den Userscreen der alte Bildschirminhalt korrekt gesichert und später auch wieder restauriert.

Mit diesem Wissen bewaffnet, galt es, den 'Fehler' zu finden... Wochen später: Nach langem Probieren schält sich eine Lösung für dieses Problem heraus.

Die Idee

Die Turbo C-Shell testet das Vorhandensein eines Großbildschirms mit dem Versuch, die Bildschirmadresse zu verändern. Wenn es gelingt, den Bildschirm probeweise auf eine andere Speicherseite umzuschalten, schaltet die Shell zwischen dem Benutzerschirm und dem eigenen Bild mittels Setscreen um; falls es jedoch nicht gelingen sollte, die Bildschirmadresse zu verändern, sichert die Shell den eigenen Bildschirm beim Start eines Programms bzw. beim Anzeigen des Userscreens in einem Puffer (wahrscheinlich mittels *vro_cpyform*).

Das Programmkonzept

Damit wird es möglich, die Shell dazu zu 'überreden', auch Hyperscreen als Großbildschirm zu akzeptieren. Ein kleiner Handler fängt den XBIOS-Aufruf *Setscreen* ab und ändert die Parameter für Physbase und Logbase in -1 ab. Damit wird dem Betriebssystem vorgegaukelt, daß diese Parameter nicht gesetzt werden sollen, was es auch prompt nicht tut.

Die TC-Shell merkt daraufhin, daß es wohl nicht möglich ist, die Bildschirmadresse zu verändern, und behandelt nun auch den SM124 wie einen 'Großbildschirm'. Damit wird es endlich möglich, die vollen 80 Zeichen in einer Zeile darzustellen; auch im Hilfefenster macht sich die größere Auflösung positiv bemerkbar.

Die Realisierung

Das Listing sollte alle Fragen zur Implementierung beantworten, da es (hoffentlich) ausreichend kommentiert ist:

Es wird mittels der Line A-Variablen [2] getestet, ob mehr Bytes pro Rasterzeile benutzt werden, als dies normalerweise der Fall ist. Nur wenn mehr Bytes benötigt werden, wird das Programm resident geladen.

Der XBIOS-Handler wird auf eine eigene Routine umgelenkt, die den Setscreen-Befehl abfängt und in die folgende Form ändert:

```
Setscreen(x,-1L,-1L);
```

Es wird aber nicht explizit getestet, ob Hyperscreen geladen ist. Damit sollte es möglich sein, auch Bigscreen o.ä. zusammen mit TC zu benutzen. Auch wird NICHT getestet, ob ein anderer Prozessor als der MC68000 im Rechner eingebaut ist. Daher ist es mit einem 68020-Board (oder gar auf dem TT) nicht lauffähig. In diesem Fall ist nämlich der Aufbau des Stacks nicht zum 68000 kompatibel...

Eine Warnung zum Schluß: Der Debugger funktioniert auch jetzt leider noch nicht, da er sehr nah (ZU NAH) an der Hardware des ST programmiert ist. Trotzdem viel Spaß mit HyperTurboC.

PP

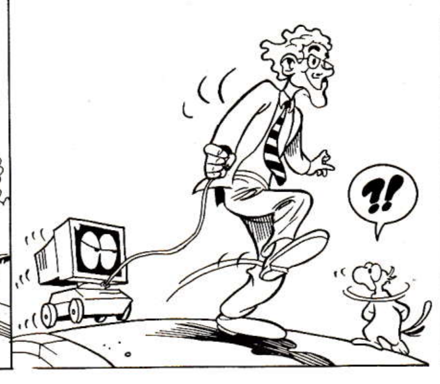
Literatur:

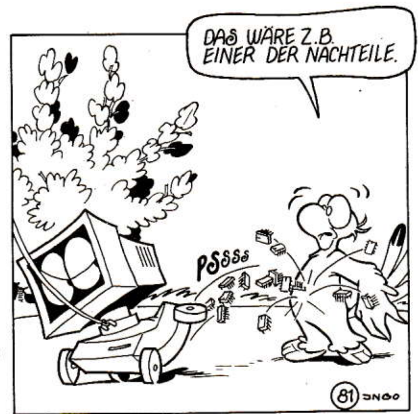
[1] ST-Magazin 5/89 S. 16ff
[2] Jankowski, Reschke, Rabich; Atari ST Profibuch; Sybex-Verlag

```
1:  ;*********************************************
2:  ;** XBIOS-Patch für Turbo C
3:  ;**
4:  ;**        Friedel van Megen
5:  ;**
6:  ;**        (c) 1991 MAXON Computer
7:  ;**
8:  ;*********************************************
9:
10: gemdos     equ   1
11: Cconws     equ   9  ;schreibe String
12: Ptermres   equ   49 ;Terminate but stay resident
13:
14: xbios      equ   14
15: Supexec    equ   38 ;exec in Supervisormode
16: Setscreen  equ   5  ;Set screenbase
17:
18: v_trp14    equ   $b8       ;Trap #14 Vektor
19:
20: p_start    bra   p_init    ;zum Start!
21:
22: ;*********************************************
23: ;** patch as patch can...
24: ;*********************************************
25: inst_vec   move.l  v_trp14,sv_trp14  ;Trap #14
                                Vektor patchen
26:            move.l  #new_trp14,v_trp14
27:            rts
28:
29: ;*********************************************
30: ;** neuer TRAP #14 handler, XBRA-tauglich,
      Kennung 'PBIT'
31: ;*********************************************
32:            dc.l  'XBRA'
33:            dc.l  'PBIT'
34: sv_trp14   dc.l  0       ;savearea für trap #14-
                               vektor
35: new_trp14  move.l  a7,a0 ;welchen Stackpointer
                               soll ich benutzen
36:            addq.l  #6,a0 ;(das SR wird in jedem
                          Fall auf dem SSP abgelegt)
37:            move.w  (sp),d1 ;Statusregister holen
38:            btst    #13,d1
39:            bne.s   in_supm ;ok, Supervisor
40:            move.l  usp,a0 ;Aufruf aus USER-Mode
41: in_supm    move.w  (a0)+,d0   ;Funktionscode
42:            cmp.w   #Setscreen,d0  ;Soll ich was
                                    machen?
43:            bne.s   end_ein  ;Nein ->
44:
45:            move.l  #-1,(a0) ;verhindern, daß die
                               Bildschirmadresse
46:            move.l  #-1,4(a0) ;geändert wird
47:
48: end_ein    move.l  sv_trp14,a0
49:            jmp     (a0)      ;dann eben nicht...
50: p_end      nop
51:
52: ;*********************************************
53: ;** initialisieren...
54: ;*********************************************
55: p_init     pea     copy_msg   ;Nur mein NAME....
56:            move.w  #Cconws,-(sp)
57:            trap    #gemdos
58:            addq.l  #6,sp
59:
60:            dc.w    $a000        ;Line A-
                             Parameterblock holen
61:            moveq.l #80,D0   ;Bytes pro Zeile
                      (bei 'normaler' Auflösung)
62:            mulu.w  (A0),D0
63:            sub.w   2(A0),D0  ;akt. Anzahl Bytes
                                pro Zeile
64:            bpl     no_big_scr ;nur die normale
                                Auflösung
65:
66:            pea     on_msg  ;Dies ist was
                             hypermäßiges
67:            move.w  #Cconws,-(sp)
68:            trap    #gemdos
69:            addq.l  #6,sp
70:            pea     inst_vec   ;nur noch Vektoren
                                patchen
71:            move.w  #Supexec,-(sp)
72:            trap    #xbios
73:            addq.l  #6,sp
74:            move.w  #0,-(sp)  ;wir bleiben
                                resident!
75:            move.l  #$100+p_end-p_start,-(sp)
76:            move.w  #Ptermres,-(sp)
77:            trap    #gemdos    ;Das war's...
78:
79: no_big_scr pea     not_on_msg
80:            move.w  #Cconws,-(sp)
81:            trap    #gemdos
82:            addq.l  #6,sp
83:            clr.w   -(sp)
84:            trap    #gemdos       ;Nicht resident
85:
86: ;*********************************************
87: ;** DATA
88: ;*********************************************
89:   data
90: copy_msg   dc.b  13,10,"Setscreen manager
                       V1.0",10,13
91:            dc.b  "by Friedel van Megen",13,10,0
92: not_on_msg dc.b  "NOT "
93: on_msg     dc.b  "INSTALLED...",13,10,0
94:   end
```









AUSGABEUMLENKUNG VIA BIOS

Jan Starzynski

WEM GEHT ES NICHT AB UND AN SO: MAN HAT EIN SCHÖNES PROGRAMM, DAS EBENSO SCHÖNE GRAFIKEN IM VEKTORFORMAT ERZEUGT UND DAHER EWIG LANGSAM IM AUSDRUCK IST?
ODER MAN PROGRAMMIERT FÜR SEINE SERIELLE SCHNITTSTELLE EINE NEUE, EXTREM SCHNELLE ÜBERTRAGUNGSROUTINE, DIE ABER NICHT GANZ RICHTIG FUNKTIONIERT. LEIDER BRAUCHT MAN ZUM TESTEN EINEN ZWEITEN RECHNER, UND DER FREUND IST GANZ SCHÖN SAUER OB DER FÜR IHN NICHT NUTZBAREN ZEIT.

Wie schön wäre es, könnte man den Drucker oder die serielle Schnittstelle durch die Tastatur oder eine Datei oder ... ersetzen. Der erfahrene ST-COMPUTER-Leser wird sagen: Dafür gibts doch die GEMDOS-Funktion $46 (*Fforce*). Stimmt ja auch, aber...

Fforce

An dieser Funktion störten mich schon lange drei Mängel:

1. Es können nur GEMDOS-Funktionen umgelenkt werden.
2. Es wird nicht nach Ein-und Ausgabefunktionen unterschieden (außer CON:).
3. Es können auch nicht alle verfügbaren Geräte umgelenkt werden (z.B. MIDI).

Was lag näher, als sich den Assembler zu nehmen und sich der Sache anzunehmen? Aber zunächst einmal zu den bekannten Sachen, sprich *Fforce*.

Diese GEMDOS-Funktion bewirkt wie bekannt das Umlenken von Geräten in andere Geräte oder Dateien. Ein kurzes Beispiel in C:

```
#define CON_IN  0
#define CON_OUT 1
#define AUX     2
#define PRN     3
/* womit wir auch schon
   alle behandelbaren
   Geräte erwähnt hätten
*/

main()
{
 int han;
 han=Fcreate("test.dat",0);
 Fforce(PRN,han);
 Cprnout('a');
 Fclose(han);
 return 0;
}
```

Es passiert im Programm nichts weiter, als daß die Datei TEST.DAT geöffnet und der Drucker dorthin umgelenkt wird. Wenn nun ein Zeichen auf den Drucker ausgegeben wird, landet es nicht auf diesem, sondern in unserer Datei, und man kann in der Datei nach Beenden des Programmes den Buchstaben 'a' bewundern. So weit, so gut. Jetzt weiß ich also, wie ich in meinem eigenen Programm den Drucker „verbiege". Das hilft mir aber beim Grafikprogramm nicht weiter. Schließlich habe ich das nicht selbst geschrieben und der Hersteller wird wohl auch kaum den Quelltext verschicken. Die Hilfe ist einfach: Diese „Geräteverbiegung" kann weitervererbt werden. Ersetzt man also das

```
Cprnout('a')
```

im Beispiel durch z.B.

```
Pexec("graphik.prg",
      command,"",0)
```

so wird GRAPHIK.PRG gestartet und die Grafik, die sonst auf dem Drucker gelandet wäre, liegt dann in unserer Datei, wo sie ja gut aufgehoben ist. Nachdem unser Programm beendet wurde, ist wieder alles in bester Ordnung, das Betriebssystem sorgt selbst dafür, daß der Drucker druckt und nicht mehr „dateit". Und wie drucke ich dann das ganze aus? Ganz einfach: vom Desktop aus die Datei zweimal anklicken und die Frage *ansehen drucken Abbruch* mit „drucken" beantworten. Dann geht die Post ab, und der Drucker wirft die schönste Grafik aufs Papier. Damit hätten wir also die einfache Lösung kurz nochmal abgeschnitten. Aber wie gesagt, das funktioniert sofort nicht mehr, wenn unser Grafikprogramm die Grafik mittels BIOS ausgibt. Dann müssen wir nach wie vor eine halbe Stunde Rechenzeit in Kauf nehmen. Und für andere Anwendungen bestehen immer noch die oben genannten Mängel. Und hier setzt mein BIOS.PRG an.

Jetzt komme ich

Was eigentlich das Programm bewirken soll, ist ja nun klar. Und weil ich *Fforce* nicht für so schlecht halte, sollte die Wirkung auch ähnlich sein. Ich habe also extra für dieses kleine Programm eine neue BIOS-Funktion implementiert, die bei mir *Bforce* heißt und die Dateiumlenkung auf BIOS-Ebene bewirkt und im Gegensatz zu *Fforce* auch noch zwischen Ein- und Ausgabe unterscheidet. Folglich gibt es drei Parameter:

1. der umzulenkende Kanal
2. wohin der Kanal umzulenken ist
3. ob Ein- oder Ausgabe gemeint ist

Um das Kapitel der Benutzung gleich ganz abzuhaken, nun noch das Schema des Aufrufs

der Funktion in C und Assembler:

```
#define Bforce(a,b,c)
bios(132,a,b,c)
Bforce(device,newdevice,
       in_out);

move.w in_out,-(sp)
move.w newdev,-(sp)
move.w device,-(sp)
move.w #132,-(sp)
trap   #13
addq.l #8,sp
```

Die Parameter haben dabei folgende Bedeutung:

in_out: 0 - Eingabe umlenken
1 - Ausgabe umlenken
newdev: BIOS- bzw. Datei-Handle des Umlenkungszieles
device: BIOS-Handle des umzuleitenden Kanales. Ist *newdev* größer oder gleich sechs, handelt es sich um ein Datei-Handle, sonst um eines der folgenden BIOS-Handles.

0 Drucker
1 serielle Schnittstelle
2 Konsole
3 MIDI
4 IKBD
5 RAW-Konsole

Datei-Handles sind automatisch größer als sechs, darüber braucht man sich also keine Sorgen zu machen. Wen jetzt die technischen Feinheiten nicht interessieren (sog. Nutzer), der kann aufhören zu lesen und anfangen, das Listing abzutippen. Allerdings sollte er sich die Warnungen am Ende des Textes noch durchlesen.

Programmierung

Dem geehrten Leser stehen natürlich noch alle anderen Wege offen. Aber so funktioniert es in etwa: Mit einer Funktion namens *install* klinke ich mich in den BIOS-Vektor (XBRA) und fange erstmal alle BIOS-Aufrufe ab. Danach werden alle mich interessierenden Funktionen ausgefiltert, als da wären

Name	Nummer
1. Bconstat	0
2. Bconin	2
3. Bconout	3
4. Bcostat	8
5. Bforce	132

Das sind alle BIOS-Funktionen, die mit Ein-und Ausgabe auf Geräte zu tun haben. 1. und 4. sind Erkundigungsfunktionen, die den Status von Geräten feststellen, 2. und 3. sind die Ein-und Ausgabefunktionen, und 5. schließlich ist hier gerade im Entstehen. Zuerst beschäftigen wir uns mit 5.

Wird diese Funktion aufgerufen, wird anhand von *device* in die Ein- oder Ausgabetabelle der Wert von *newdev* eingetragen. *device* wird dabei als Offset in der Tabelle verwendet.

Wird nun allerdings 1., 2., 3. oder 4. erkannt, liegt als Funktionsparameter auch immer die *device*-Nummer auf dem Stack. Diese wird vom neuen BIOS-Handler als Offset in die Ein- oder Ausgabetabelle verwendet und die dort stehende Nummer anstelle des alten Wertes auf den Stack gelegt. Kurz gesagt, wird die Funktionsnummer ausgetauscht. Danach wird der BIOS-Aufruf weitergeleitet an den ursprünglichen BIOS-Handler, der dann ganz unbekümmert mit dem untergemogelten Gerät weitermacht, als wäre nichts passiert. Das Programm, das den Aufruf tätigt, bekommt auch nichts von der kleinen Manipulation mit.

So wie bis hier dargestellt, funktioniert das ganze aber nur für Umleitungen innerhalb der sechs BIOS-Kanäle. In Dateien kann man damit aber noch nichts schreiben. Da aber Datei-Handles gut zu erkennen sind (>=6), wird in diesem Fall wie folgt vorgegangen:

Umleitung in Dateien

Für Ein- oder Ausgabefunktionen wird die in der Tabelle gefundene neue Gerätenummer als Datei-Handle erkannt und zur weiteren Bearbeitung das GEMDOS genutzt. Der zur Ausgabe vorgesehene Wert wird also in einen Ein-Byte-Puffer übertragen und mittels *Fwrite* in die ausgewählte Datei ausgegeben. Bei der Eingabe wird der einzulesende Wert mittels *Fread* in den Puffer eingelesen und danach in D0 übertragen, wo das BIOS alle Funktionswerte zurückgibt. Die Erkundigungsfunktionen sind in diesem Fall nur Dummy-Funktionen, da sie immer wahr zurückgeben. Es gibt also hier einen „Rückschritt" vom BIOS zum GEMDOS, der allerdings nicht ganz problemlos funktioniert, da das GEMDOS auf das BIOS zurückgreift. Wenn nämlich ein BIOS-Aufruf des GEMDOS erfolgt, und dieses wiederum das GEMDOS konsultiert, werden die internen Speicher des GEMDOS gnadenlos überschrieben, und man kriegt das große Grübeln, wo plötzlich die ganzen Bomben herkommen. Aber man kann sich ja helfen. Die Lösung ist zwar nicht so ganz elegant, funktioniert aber. Es wird nämlich auch noch der GEMDOS-Vektor überwacht, und sobald eine Funktion (außer *Pexec*) angesprungen wird, ignoriert unsere BIOS-Routine alle Umleitungen in Dateien, so daß das GEMDOS ungestört arbeiten kann. Dadurch ist zwar die Umleitung nicht mehr ganz so mächtig, aber dafür sicherer.

Das war's im Prinzip auch schon. Als Zugabe ist noch ein kleines Programm dabei, das die BIOS-Routine resident im Speicher ablegt. Und zum Schluß nun noch die versprochenen Warnungen.

Achtung!

Als wichtigstes und allgemeinstes: am BIOS führt (fast) kein Weg vorbei. Eine Umlenkung von Kanälen wirkt sich also überall aus, selbst bei solchen Sachen wie der Fileselectbox! Also Vorsicht bei der Anwendung, insbesondere bei Umlenkung der Standardeingabe. Des weiteren ist zu beachten, daß nicht alle BIOS-Funktionen für alle Geräte implementiert sind. Die hier vorgestellte Routine kümmert sich um solche Feinheiten nicht, aber die Original-BIOS-Fehlermeldungen kommen zurück. Das gilt nicht bei der Umlenkung in Dateien. Man sollte also immer sehen, daß die Dateiarbeit reibungsfrei ablaufen kann. Des weiteren macht das Betriebssystem die Veränderung der Kanäle nicht selbständig wieder rückgängig, sondern der Nutzer muß das selbst machen [mit *Bforce (device,device,in_out)*]. Sonst bleibt alles bestehen bis zum nächsten Reset. Die Funktionen install und exstall, sollte sie jemand in einem eigenen Programm nutzen wollen, müssen im Supervisormodus aufgerufen werden. Ich hoffe, ich habe nun alle Gefahren ertappt, die auf den Nutzer warten, wenn ich natürlich dafür auch keine Garantie übernehmen kann. Getestet wurden die Routinen übrigens auf einem STE mit 1 MByte.

```
 1:            GLOBAL install,exstall
 2:
 3:  bios      EQU $B4                  ;adr des bios-
 4:  ;                                   vektors
 5:  gemdos    EQU $84                  ;adr des gemdos-
 6:  ;                                   vektors
 7:
 8:
 9:  anfang:   bra      schluss
10:            DC.B "XBRABFOR"
11:  oldbios:DC.L 1
12:  mybios:
13:            movem.l D0-D1/A0-A1,-(SP)
14:            lea      22(SP),A0      ;funktionsnum-
15:  ;                                  mer auf superstack
16:            move.w   -6(A0),D0      ;statusregister
17:            btst     #13,D0         ;alter Status
18:  ;                                  == superviser?
19:            bne.s    aktiv          ;nein->zu aktiv
20:            move     USP,A0         ;sonst usp
21:  ;                                  laden
22:  aktiv:
23:            move.w   (A0),D0        ;funktionsnum-
24:  ;                                  mer laden
25:            cmpi.w   #132,D0        ;meine umlenk-
```
→

```
26: ;                               funktion?
27:          beq      newnumb      ;ja->
28: ;                               neuumlenkung
29:          cmpi.w   #1,D0        ;nein, Bconin?
30:          beq.s    input        ;ja ->
31: ;                               inputfunktion
32:          cmp.w    #2,D0        ;Bconstat?
33:          beq.s    input        ;ja
34:          cmpi.w   #3,D0        ;Bconout?
35:          beq.s    outputs      ;ja
36:          cmpi.w   #8,D0        ;Bcostat?
37:          beq.s    outputs      ;ja
38: back:       ;eigene arbeit erledigt
39:          movem.l  (SP)+,D0-D1/A0-A1 ;weiter im
40: ;                                   bios
41:          move.l   oldbios(PC),-(SP)
42:          rts
43:
44: input:
45:          move.w   2(A0),D0     ;device in d0
46:          add.w    D0,D0        ;2* wegen word
47:          lea      itab(PC),A1  ;inputtab in a1
48:          cmpi.w   #6,0(A1,D0.w) ;wert in tab<6?
49:          bcs.s    biosin
50: ;                               ja->
51: ;                               umlenkung bleibt
52: ;                               im bios
53:
54: ;   ab hier umlenkung in gemdos
55:          tst.b    isdos        ;aufruf aus
56: ;                               gemdos?
57:          bne.s    back         ;ja, dann weiter
58:          cmpi.w   #1,(A0)      ;Bconstat?
59:          beq.s    info
60: ;                               ja, dann wie
61: ;                               Bcostat
62:
63: dosin:   move.w   0(A1,D0.w),D0
64: ;                               nein, gemdos
65: ;                               lesen:
66: ;                               device in d0
67:          pea      buf(PC)      ;alles klar-
68: ;                               machen für
69: ;                               Fread
70:          move.l   #1,-(SP)
71:          move.w   D0,-(SP)
72:          move.w   #$3F,-(SP)
73:          trap     #1
74:          lea      12(SP),SP
75: inend:   movem.l  (SP)+,D0-D1/A0-A1
76:          moveq    #0,D0        ;wegen long-
77: ;                               rückgabe
78: ;                               löschen
79:          move.b   buf(PC),D0   ;ergebnis in d0
80: ;                               bringen
81:          rte
82:
83: ;ende der gemdosarbeit
84:
85: biosin:  move.w   0(A1,D0.w),2(A0) ;funktions-
86: ;                                   nummer aus
87: ;                                   tabelle
88: ;                                   übernehmen
89:          bra.s    back         ;weiter
90: ;                               im bios
91:
92:
93: outputs:
94:          move.w   2(A0),D0     ;device in d0
95:          add.w    D0,D0        ;2* wegen word
96:          lea      otab(PC),A1  ;inputtab in a1
97:          cmpi.w   #6,0(A1,D0.w) ;wert in tab<6?
98:          bcs.s    biosout      ;ja-umlenkung
99: ;                               bleibt im bios
100:         tst.b    isdos        ;aufruf aus
101: ;                              gemdos?
102:         bne.s    back         ;ja, dann weiter
103:         cmpi.w   #8,(A0)      ;Bcostat?
104:         bne.s    dosout       ;nein, dann
105: ;                              raus damit
106: info:    movem.l  (SP)+,D0-D1/A0-A1 ;register
107: ;                                  zurück
108:         move.l   #-1,D0       ;wert
109: ;                              zurückgeben
110:         rte
111:
112: dosout: move.w   0(A1,D0.w),D0 ;gemdos:
113: ;                              newdev in d0
114:         move.w   4(A0),D1     ;zeichen-
115:         move.b   D1,buf       ;argument in
116: ;                              buffer bringen
117:         pea      buf(PC)      ;alles klar-
118:         move.l   #1,-(SP)     ;machen für
119:         move.w   D0,-(SP)     ;Fwrite
120:         move.w   #$40,-(SP)
121:         trap     #1
122:         lea      12(SP),SP
123:         bra.s    exit
124:
125: biosout:move.w   0(A1,D0.w),2(A0) ;funktions-
126: ;                                  nummer aus
127: ;                                  tabelle
128: ;                                  übernehmen
129:         bra      back         ;weiter im bios
130:
131: newnumb:move.w   2(A0),D0     ;device
132:         add.w    D0,D0        ;2* wegen word
133:         move.w   4(A0),D1     ;neue device-
134: ;                              nummer
135:         lea      itab(PC),A1  ;tabelle für
136: ;                              input
137:         tst.w    6(A0)        ;input
138: ;                              gewünscht?
139:         beq.s    umleit       ;ja->dann los
140:         lea      otab(PC),A1  ;sonst output-
141: ;                              tabelle
142: umleit:
143:         move.w   D1,0(A1,D0.w) ;neue num-
144: ;                              mer eintragen
145: exit:    movem.l  (SP)+,D0-D1/A0-A1
146:         rte
147:
148: install::
149:         move.l   A0,-(SP)
150:         movea.l  bios.w,A0
151:         cmpi.l   #"BFOR",-8(A0) ;schon
152: ;                              installiert?
153:         beq.s    fast_ende
154:         move.l   A0,oldbios   ;alten bios
155: ;                              retten
156:         move.l   #mybios,bios.w ;neuen ein-
157: ;                              setzen
158: fast_ende:            ;nochmal das ganze mit gemdos
159:         movea.l  gemdos.w,A0
160:         cmpi.l   #"BFOR",-8(A0)
161:         beq.s    ende
162:         move.l   A0,dosvek
163:         move.l   #mydos,gemdos.w
164: ende:    movea.l  (SP)+,A0     ;und
165:         rts                   ;zurück
166:
167:
168:
169: ;hier kann man alles wieder rückgängig machen
170:
171: exstall::cmpi.l  #mybios,bios.w
172:         bne.s    fast_bye
173:         move.l   oldbios(PC),bios.w
174: fast_bye:
175:         cmpi.l   #mydos,gemdos.w
176:         bne.s    bye
177:         move.l   dosvek(PC),gemdos.w
178: bye:     rts
179:
180:
181:
182: ;einklinken in GEMDOS-Vektor
183:
184:         DC.B "XBRABFOR"
185: dosvek: DC.L 1
186: mydos:
187:         movem.l  D0/A0,-(SP)
188:         move.w   14(SP),D0    ;funktionsnummer
189:         btst     #5,8(SP)     ;statusregister
190:         bne.s    ok           ;supervisor
191:         move     USP,A0       ;sonst user
192:         move.w   (A0),D0      ;funktionsnummer
193: ok:
194:         cmp.w    #$4B,D0      ;pexec?
195:         beq.s    dosend       ;ja, dann ok
196:         move.l   10(SP),saveptr ;sonst
197:         move.l   #dosret,10(SP) ;einklinken
198:         st       isdos        ;bin in gemdos
199: dosend: movem.l  (SP)+,D0/A0  ;weiter im
200:         move.l   dosvek(PC),-(SP) ;gemdos
201:         rts
202: dosret: clr.b    isdos        ;zurück aus
203:         move.l   saveptr(PC),-(SP) ;gemdos
```

→

```
204:           rts
205:
206:
207:  saveptr:DC.L 1
208:  itab:                                  ;tabelle der
209:           DC.W 0                        ;eingabe-
210:           DC.W 1                        ;leitung
211:           DC.W 2
212:           DC.W 3
213:           DC.W 4
214:           DC.W 5
215:
216:  otab:    DC.W 0                        ;und der
217:           DC.W 1                        ;Ausgabe-
218:           DC.W 2                        ;umleitung
219:           DC.W 3
220:           DC.W 4
221:           DC.W 5
```

```
222:
223:  buf:
224:           DS.B 1                    ;puffer für
225:  ;                                   Fread
226:  isdos:   DS.B 1                    ;semaphore für
227:  ;                                   gemdos-erkennung
228:           EVEN
229:  ;ein kleines installationsprogramm
230:  schluss:pea      install(PC)    ;Supexec
231:           move.w   #38,-(SP)
232:           trap     #14
233:           addq.l   #6,SP
234:           clr.w    -(SP)          ;Ptermres
235:           move.l   #(256+schluss-anfang),-(SP)
236:           move.w   #$31,-(SP)
237:           trap     #1
238:
239:           END
```

PROZEDUR FÜR SCHÖNERE KREISE

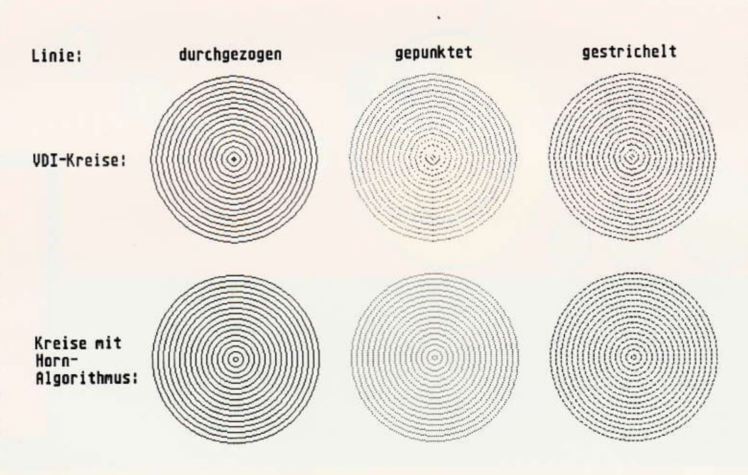
Andreas Hollmann

EINEN ECHTEN KREIS AUF EINEM QUADRATISCHEN RASTER, WIE ES BEI BILDSCHIRM- UND DRUCKERAUSGABE VORLIEGT, ZU ZEICHNEN, IST UNMÖGLICH. ES BESTEHT NUR DIE MÖGLICHKEIT, DIE EINZELNEN PUNKTE DES KREISUMFANGS IM RASTER SO ZU SETZEN, DASS DIE ABWEICHUNG VON DER IDEALEN LINIE MÖGLICHST GERING IST. ZUR LÖSUNG DIESES PROBLEMS WURDEN IM LAUF DER ZEIT VERSCHIEDENE ALGORITHMEN BENUTZT. DER ALGORITHMUS, DER IN DER VDI-KREISROUTINE VERWENDUNG GEFUNDEN HAT, BESITZT ZWEI MERKMALE, DIE SOFORT INS AUGE STECHEN: DIE KREISE WERDEN UNHEIMLICH SCHNELL UND UNHEIMLICH HÄSSLICH GEZEICHNET.

Leider habe ich keine Informationen über den benutzten Algorithmus, aber, betrachtet man den Kreisbogen bei bestimmten Radien, so drängt sich die Vermutung auf, daß in Wirklichkeit Polygone gezeichnet werden. Das würde auch die hohe Geschwindigkeit erklären, denn ein paar Polygonecken sind schneller zu berechnen als jeder einzelne Punkt auf dem Kreisumfang.

Die durchgezogenen Linien der VDI-Darstellung sind schon schlimm genug, aber bei Verwendung verschiedener Linienmuster packt einen das Grauen. Genau diese Anforderung (die Kreise, nicht das Grauen) stellte aber eine von mir zu programmierende Routine: es sollten u.a. konzentrische Kreise mit einem gepunkteten Linienmuster ausgegeben werden.

Es blieb also nichts anderes übrig, als bessere Kreise durch das Setzen einzelner Pixel zu generieren. Der folgende Algorithmus basiert auf Horns Algorithmus und zeichnet sich dadurch aus, daß zur Berechnung nur Additionen, Subtraktionen und binäre Schiebeoperationen benutzt werden. Wer also eine höhere Geschwindigkeit benötigt, dem sollte es nicht allzu schwer fallen, die Kreisroutine in Assembler zu programmieren. Auch die Konvertierung von GFA-BASIC in andere Sprachen ist ohne weiteres möglich. Für Konvertierer sei noch angemerkt, daß der Befehl PRED den um eins verminderten, und der Befehl SUCC den um eins erhöhten Wert der übergebenen Variablen zurückliefert.



Hier der Vergleich: Alle Kreise wirken rund - oberflächlich betrachtet; doch beim VDI-Kreis hier und da Schmutzränder. Bei der Verwendung von gemusterten Linien ist besonders die VDI-Interpretation der kleinen Kreise stark gewöhnungsbedürftig.

Der Routine werden die x- und y-Koordinaten des Kreismittelpunktes, der Radius und ein 16-Bit-Wort, welches das Linienmuster bestimmt, übergeben. Zum Setzen der einzelnen Punkte wird der PLOT-Befehl benutzt. Dadurch ist garantiert, daß das VDI-Clipping-Rechteck berücksichtigt wird und auf Bildschirmen mit anderer Auflösung keine Probleme auftreten.

Der Kreis wird in acht Oktanten unterteilt, deren Kreisbögen nacheinander gezeichnet werden. Vor dem Setzen eines Punktes wird geprüft, ob das Bit 0 im Linienmuster-Wort gesetzt ist. Danach wird das Linienmuster einfach durch einen ROL&-Befehl weiterrotiert, man spart sich dadurch aufwendige Bit-Zählerei.

Das Ergebnis läßt sich in der abgebildeten Hardcopy begutachten. Beim den Kreisen mit durchgezogener Linie fällt auf, daß selbst Kreise mit einem Radius von 2 Pixeln (kleinster Kreis) noch kreisähnliche Objekte ergeben, wogegen man bei VDI-'Kreisen' mit derart kleinen Radien schon Sterne sieht.

Beim gepunkteten VDI-'Kreis' kleben teilweise einige Pixel in Gruppen zusammen, als Ausgleich dafür läßt VDI aber an anderen Stellen etwas größere Lücken ... Horns Algorithmus zeigt solche Effekte nicht und setzt die Punkte in gleichmäßigen Abständen.

Literatur:
Computer Graphics and Image Processing
1979 Marek Doros
Incremental Circle Generator

```
 1:  ' ***********************************************
 2:  ' * Programm: inkrementaler Kreisgenerator
 3:  ' * Autor:     Andreas Hollmann
 4:  ' * Sprache:   GFA-BASIC
 5:  ' * Copyright 1991 MAXON Computer GmbH
 6:  ' ***********************************************
 7:  '
 8:  ' Linienmuster definieren:
 9:  CARD{V:type&}=&X1111111111111111   ! durchgezogen
10:  ' CARD{V:type&}=&X1010101010101010 ! gepunktet
11:  ' CARD{V:type&}=&X1110111011101110 ! gestrichelt
12:  '
13:  FOR r&=2 TO 80 STEP 5        ! konzentrische Kreise
14:    DEFLINE 1     ! weil DEFLINE auch auf PLOT wirkt
15:    circle(160,199,r&,type&)        ! echter Kreis
16:    DEFLINE -CARD{V:type&}          !
17:    CIRCLE 480,199,r&               ! VDI-'Kreis'
18:  NEXT r&
19:  '
20:  END
21:  '
22:  PROCEDURE circle(x0&,y0&,r&,type&)
23:    ' Parameter:
24:    ' x0&,y0&   = Kreismittelpunkt-Koordinaten
25:    ' r&        = Kreisradius
26:    ' type&     = Linienmuster (16 Bits)
27:    '
28:    LOCAL octant&,x&,y&
29:    '
30:    x&=r&
31:    '
32:    FOR octant&=1 TO 8
33:      IF ODD(octant&)          ! Oktant 1,3,5,7
34:        SUB r&,SHL&(x&,1)
35:        DO UNTIL y&>x&
36:          IF BTST(type&,0)   ! Muster-Bit gesetzt
37:            SELECT octant&   ! Punkt setzen
38:            CASE 1
39:              PLOT ADD(x0&,x&),ADD(y0&,y&)
40:            CASE 3
41:              PLOT SUB(x0&,y&),ADD(y0&,x&)
42:            CASE 5
43:              PLOT SUB(x0&,x&),SUB(y0&,y&)
44:            CASE 7
45:              PLOT ADD(x0&,y&),SUB(y0&,x&)
46:            ENDSELECT
47:          ENDIF
48:          type&=ROR&(type&,1)   ! Muster
                                     weitersetzen
49:          ADD r&,SUCC(SHL&(y&,1))
50:          INC y&
51:          IF r&>=0
52:            SUB r&,SUB(SHL&(x&,1),2)
53:            DEC x&
54:          ENDIF
55:        LOOP
56:      ELSE                       ! Oktant 2,4,6,8
57:        ADD r&,SHL&(x&,1)
58:        DO UNTIL y&<0
59:          SUB r&,PRED(SHL&(y&,1))
60:          DEC y&
61:          IF r&<0
62:            ADD r&,ADD(SHL&(x&,1),2)
63:            INC x&
64:          ENDIF
65:          IF x&<>y&  ! Diagonalen-Linien vermeiden
66:            IF BTST(type&,0) ! Muster-Bit gesetzt
67:              SELECT octant& ! Punkt setzen
68:              CASE 2
69:                PLOT ADD(x0&,y&),ADD(y0&,x&)
70:              CASE 4
71:                PLOT SUB(x0&,x&),ADD(y0&,y&)
72:              CASE 6
73:                PLOT SUB(x0&,y&),SUB(y0&,x&)
74:              CASE 8
75:                PLOT ADD(x0&,x&),SUB(y0&,y&)
76:              ENDSELECT
77:            ENDIF
78:            type&=ROR&(type&,1) ! Muster
                                     weitersetzen
79:          ENDIF
80:        LOOP
81:        type&=ROR&(type&,1) ! Muster weitersetzen
82:      ENDIF
83:      '
84:    NEXT octant&
85:    '
86:  RETURN
```

DIE SCHNELLERE DIALOGBOX

Ulrich Mast

Als ST-Besitzer gibt es jede Menge Möglichkeiten mit ein paar Tricks noch ein wenig mehr aus dem Rechner herauszuholen. Hier will ich Ihnen etwas zum Thema „Dialogbox und Geschwindigkeit" erzählen.

Grundsätzlich ist zu bedenken, daß eine Dialogbox umso schneller aufgebaut ist, je weniger zu zeichnen ist. Aus diesem Grund vermeide ich Füllmuster und unnötige Umrandungen innerhalb der Box. Da der Mensch (hierzulande zumindest) von links nach rechts und von oben nach unten liest, sollte sich eine Dialogbox genau in dieser Reihenfolge aufbauen. Der eigentliche Aufbau wird dadurch natürlich nicht schneller, dem Benutzer erscheint es jedoch so, da er sofort links oben zu lesen beginnen kann. Um diese Ordnung zu erreichen, gibt es in jedem Resource Construction Set eine Funktion zum Sortieren der Box. Da man die einzelnen Elemente aus jeder Programmiersprache mit Namen ansprechen kann, dürfte auch ein nachträgliches Sortieren keine Schwierigkeiten machen.

Daß man bei der Textausgabe unter GEM (mit *v_gtext()*) einen beträchtlichen Geschwindigkeitszuwachs erzielen kann, indem man beim Systemzeichensatz (bzw. bei allen 8 Bit breiten Zeichensätzen) mit der Ausgabe an Byte-Grenze beginnt, dürfte mittlerweile ein alter Hut sein. Als ich neulich mehrere Dialogboxen mit Hilfstexten in ein Programm einbaute, war ich über die unterschiedliche Zeichengeschwindigkeit der fast gleich großen Boxen doch sehr überrascht. Nach genauerer Untersuchung hatte ich des Rätsels Lösung gefunden: Die 'schnellere' Dialogbox hatte eine geradzahlige Zeichenbreite (Bild 1). Die AES-Funktion *form_center()* kann die Box dann so zentrieren, daß deren Objekte an Byte-Grenze plaziert werden. Intern werden die Strings dann mit *v_gtext()* auch an Byte-Grenze gezeichnet. Ist die Box aber um ein Zeichen breiter (Bild 2), werden alle Objekte an Byte-Grenze+4 gezeichnet. Dies ist natürlich langsamer.

Der Trick funktioniert natürlich nur, wenn die Objekte innerhalb der Dialogbox zeichenweise ausgerichtet sind. Bei manchen Resource Construction Sets kann man das umgehen und die Objekte beliebig positionieren. Das sollte nach Möglichkeit vermieden werden.

Der Geschwindigkeitszuwachs gilt hauptsächlich für Objekte des Typs STRING. TEXT-Objekte werden unter Umständen noch zentriert. Um hier auf Byte-Grenze zu kommen, müssen die Breite des Objekts und die String-Länge geradzahlig sein (oder beides ungeradzahlig). Beim Konstruieren einer Box sollten die Buchstaben in x-Richtung immer gleich ausgerichtet sein.

Neuerdings kommt es immer mehr in Mode, sich eigene Objekte (z.B. Mac-Buttons) zu definieren (ja, hab' ich auch schon gemacht...). Auch hierbei sollte man das oben Gesagte berücksichtigen. Bei meinen Mac-Buttons wird zuerst der Knopf gezeichnet und dann rechts daneben der Text. Um die x-Koordinate für *v_gtext()* zu bestimmen, sollte man zur x-Koordinate, die man vom GEM erhalten hat, ein Vielfaches von 8 addieren. Wurde die Box auf Byte-Grenze zentriert, werden auch die neuen Objekte mit Maximalgeschwindigkeit gezeichnet.

Wenn wir schon bei den benutzerdefinierten Objekten sind: In letzter Zeit mußte ich überall lesen, daß man mit dem LASER C-Compiler keine benutzerdefinierten Objekte erstellen könne. Vielleicht mache ich ja etwas falsch, aber bei mir funktioniert das einwandfrei. Lokale Variablen werden auf dem Stack abgelegt und sind so ganz normal zugänglich. Globale Variable werden direkt adressiert und nicht wie früher beim MEGAMAX C indirekt über ein Adreßregister. Deren Verwendung ist also kein Problem mehr.

Bei der Dialogboxverwaltung kann man einiges an Zeit gewinnen, wenn man die AES-Funktionen *form_dial()* nicht benutzt und den Hintergrund selbst (und zwar richtig) rettet. Dazu muß man aus der Größe des zu sichernden Bereichs und der Anzahl der Bitplanes den Speicherbedarf berechnen, Speicher anfordern und den Bildschirmbereich mit Hilfe von *vro_cpyfm()* dort hinkopieren. „Ja, ja, weiß ich doch schon alles", werden Sie nun zu recht sagen. Aber wissen Sie auch, wie man die *vro_cpyfm()*-Funktion so richtig ausnutzt?

Also mal angenommen, wir haben an den Koordinaten [555,473] (Großbildschirm) ein Rechteck mit der Ausdehnung [217,59]. Die Werte spielen dabei absolut keine Rolle. Das ist also unser Quellrechteck. Kopiert wird es nun nach [0,0] in unseren Speicher (natürlich wieder mit der Ausdehnung [217,59]. Was ist daran falsch? Um ehrlich zu sein: nichts, aber ... es geht schneller. Betrachten wir nun nur die x-Koordinaten. Jede Linie wird von der Position 555 auf 0 gebracht. Da 555 nicht im geringsten durch 16 teilbar ist, 0 aber sehr wohl, muß der komplette Block Wort für Wort geshiftet werden. Das kostet extrem viel Zeit (analog der Text-Shifterei oben). Und da beim Restaurieren der ganze Block wieder zurückgeshiftet werden muß, haben wir gleich zweimal sinnlos Zeit verbraten.

Da 555 mod 16 = 11 ist, verschieben wir das Rechteck besser nach [11,0]. Die Shifterei entfällt, und das Kopieren wird sichtbar schneller. Man muß natürlich noch berücksichtigen, daß man 1 Wort pro Zeile mehr Speicher braucht, aber das fällt kaum ins Gewicht.

Kommen wir nun zum Listing. Das Programm ist bewußt einfach gehalten, man bekommt als Ergebnis nur die Meßwerte geliefert. Von der Kopiererei sieht man nichts. Sie dürfen also nich enttäuscht sein...

In der Funktion *main()* melden wir uns beim VDI an. In *test()* holen wir uns die Koordinaten des Desktop-Hintergrunds (Window 0). Diesen Bereich wollen wir verschieben. Dann werden die beiden MFDBs ausgefüllt und Speicher zum Verschieben angefordert.

Wenn wir diesen erhalten, werden die Maus ausgeschaltet (stört nur) und die aktuellen Daten des Bildschirms ausgegeben (der Betrachter will ja auch was sehen). Dann werden 2 Tests durchgeführt (und zwar jeweils 16mal für alle möglichen Offsets zur Wortgrenze): Test 1 verschiebt immer zur Wortgrenze. Bei Test 2 dagegen verschieben wir an die x-Koordinate mod 16. Die Funktion *do_test()* führt die Messung durch und gibt die Zeiten aus. Das Rechteck wird jeweils 10mal vom Bildschirm in den Speicher kopiert. Da man beim Zurückkopieren sowieso nichts sehen würde, weil sich der Bildschirm in der Zwischenzeit nicht verändert hat, wird dies unterlassen. *copy()* übersetzt die Parameter in eine dem VDI verständliche Form und ruft dieses dann auf. *get_timer()* liest den 200 Hz-Systemtakt aus.

PP

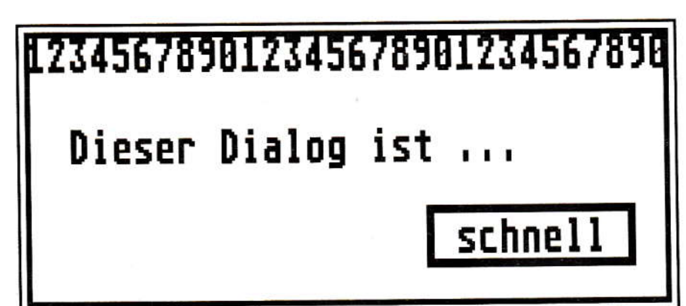


Bild 1: 30 Zeichen breite Box

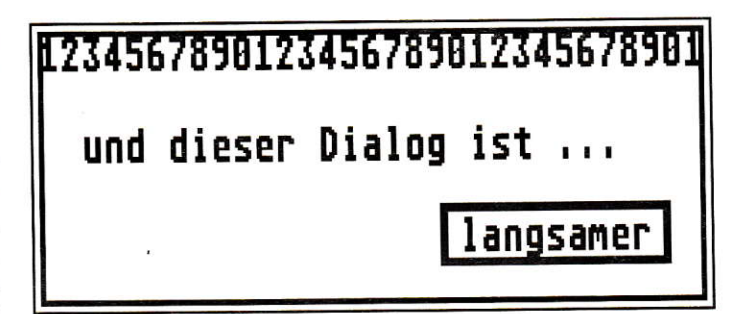


Bild 2: 31 Zeichen breite Box

vro_cpyfm()-Geschwindigkeitsdemo

Auflösung: 639 x 399
Farbebenen: 1
Rechteck: [0,19] [623,399]
Speicher: 30480

Test 1: von	nach	Zeit (ms)	Test 2: von	nach	Zeit (ms)
0	0	830	0	0	805
1	0	1675	1	1	825
2	0	1680	2	2	825
3	0	1755	3	3	870
4	0	1760	4	4	825
5	0	1830	5	5	820
6	0	1830	6	6	820
7	0	1905	7	7	820
8	0	1905	8	8	825
9	0	1900	9	9	825
10	0	1830	10	10	825
11	0	1830	11	11	825
12	0	1755	12	12	825
13	0	1750	13	13	825
14	0	1675	14	14	825
15	0	1675	15	15	825

gemessen auf ATARI ST, 8 MHz, TOS 1.4, kein Blitter, Auflösung: 640x400

nur 6 Text-Objekte beim Blättern wieder aufbauen

```
 1: /***********************************************/
 2: /*                                             */
 3: /*   vro_cpfm.c                                */
 4: /*   vro_cpyfm() Speed Demo                    */
 5: /*                                             */
 6: /*   Programmiert mit LASER C V2.1 von         */
 7: /*                                             */
 8: /*   Ulrich Mast                               */
 9: /*                                             */
10: /*   (c) 1991 MAXON Computer                   */
11: /*                                             */
12: /***********************************************/
13: #include   <osbind.h>
14: #include   <gemdefs.h>
15:
16: #define    HZ_200        0x4BA
17:
18: int     contrl[12],
19:        intin[128],intout[128],
20:        ptsin[128],ptsout[128];
21:
22: int     vdi_handle,
23:        vdi_planes,
24:        vdi_w,
25:        vdi_h;
26: int     work_out[57],
27:        work_in[]={1,1,1,1,1,1,1,1,1,1,2};
28: MFDB screen,mem;
29:
30:
31: long
32: get_timer()
33: {
34:   long  stack;
35:   long  timer;
36:
37:   stack=Super(0L);
38:   timer=*(long*)HZ_200;
39:   Super(stack);
40:   return(timer);
41: }
42:
43: copy(x,y,w,h,diff)
44: int      x,y,w,h,diff;
45: {
46:   int       xy[8];
47:
48:   xy[0]=x;          /* Quelle   */
49:   xy[1]=y;
50:   xy[2]=x+w-1;
51:   xy[3]=y+h-1;
52:   xy[4]=diff;          /* Ziel     */
53:   xy[5]=0;
54:   xy[6]=w-1+diff;
```

→

```
55:      xy[7]=h-1;
56:
57:      vro_cpyfm(vdi_handle,3,xy,&screen,&mem);
58:  }
59:
60:  do_test(x,y,w,h,diff)
61:  int  x,y,w,h,diff;
62:  {
63:      long  timer;
64:      int       i;
65:
66:      timer=get_timer();
67:      for(i=0;i<10;i++)
68:         copy(x,y,w,h,diff);
69:      printf("%3d %4d   %ld\n",
70:            x,diff,(get_timer()-timer)*5L);
71:  }
72:
73:  test()
74:  {
75:      int       x,y,w,h;
76:      long  len;
77:      int       i;
78:
79:      wind_get(0,WF_WORKXYWH,&x,&y,&w,&h);
80:      w-=16;
81:
82:      screen.fd_addr=0L;
83:
84:      mem.fd_nplanes=vdi_planes;
85:      mem.fd_wdwidth=(w+31)/16;
86:      mem.fd_w=mem.fd_wdwidth*16;
87:      mem.fd_h=h;
88:      mem.fd_stand=0;
89:
90:      len=2L;
91:      len*=(long)mem.fd_wdwidth;
92:      len*=(long)mem.fd_h;
93:      len*=(long)mem.fd_nplanes;
94:
95:      mem.fd_addr=Malloc(len);
96:      if(!mem.fd_addr)
97:      {
98:         form_alert(1,
99:         "[3][ Nicht genug Speicher ! ][Oha]");
100:        return;
101:     }
102:
103:     graf_mouse(M_OFF,0L);
104:
105:     printf("\nvro_cpyfm() Speed Demo\n\n");
106:     printf("Auflösung:  %d x %d\n",vdi_w,vdi_h);
107:     printf("Farbebenen: %d\n",vdi_planes);
108:     printf("Rechteck:       [%d,%d] [%d,%d]\n",
109:                x,y,x+w-1,y+h-1);
110:     printf("Speicher:       %ld\n\n",len);
111:
112:     printf("Test 1:\n\n");
113:     printf("von nach   Zeit (ms)\n");
114:     for(i=0;i<16;i++)
115:        do_test(x+i,y,w,h,0);
116:     printf("Taste...\n\n");
117:     Cconin();
118:
119:     printf("Test 2:\n\n");
120:     printf("von nach    Zeit  (ms)\n");
121:     for(i=0;i<16;i++)
122:        do_test(x+i,y,w,h,(x+i)%16);
123:
124:     printf("Taste...\n\n");
125:     Cconin();
126:
127:     graf_mouse(M_ON,0L);
128:
129:     Mfree(mem.fd_addr);
130: }
131:
132: main(argc,argv)
133: int      argc;
134: char *argv[];
135: {
136:     int       d;
137:
138:     appl_init();
139:     graf_mouse(ARROW,0L);
140:
141:     vdi_handle=graf_handle(&d,&d,&d,&d);
142:     v_opnvwk(work_in,&vdi_handle,work_out);
143:     if(vdi_handle)
144:     {
145:        vdi_w=work_out[0];
146:        vdi_h=work_out[1];
147:
148:        vq_extnd(vdi_handle,1,work_out);
149:        vdi_planes=work_out[4];
150:
151:        test();
152:
153:        v_clsvwk(vdi_handle);
154:     }
155:     else
156:        form_alert(3,
157:        "[1][ Kann vWK nich öffnen ! ][Hmnm]");
158:
159:     appl_exit();
160: }
```

DEN BILDSCHIRM-SPEICHER IM GRIFF

Mario Srowig

DASS DER BILDSCHIRM EIN SEHR WICHTIGES AUSGABEGERÄT IST, WIRD WOHL NIEMAND BESTREITEN KÖNNEN. AUS DIESEM GRUND SOLLTE SICH JEDER ERNSTHAFTE PROGRAMMIERER DAMIT AUSEINANDERSETZEN, WIE UND WO DER COMPUTER DIE BILDINFORMATION ABSPEICHERT. EIN SOLCHES WISSEN IST SPÄTESTENS DANN ERFORDERLICH, WENN MAN PROGRAMME ENTWIKKELN WILL, DIE MIT GRAFISCHEN SPEZIALEFFEKTEN AUSGESTATTET WERDEN ODER DEREN ARBEITSBEREICH ÜBER DIE GRENZEN DES GERADE AUF DEM BILDSCHIRM SICHTBAREN BEREICHS HINAUSGEHEN SOLLEN.

Wie ist es zum Beispiel möglich, daß bei Computerspielen Landschaften sofort aus dem Nichts auftauchen, ohne daß man auf lange Berechnungen oder auf einen Diskettenzugriff warten muß? Oder wie wie ist es möglich, zwei Bildschirme zu einer Einheit zusammenzufassen, um beispielsweise mit einem Malprogramm eine komplette DIN A4-Seite zu bearbeiten? Die Antwort auf dieseund weitere Fragen bietet dieser Artikel.

Speichern der Bildinformation

Alles, was auf dem Bildschirm zu sehen ist, ist irgendwo im RAM des Atari ST gespeichert. Dabei ist es dem Computer völlig egal, ob es sich um Grafik oder reine Textinformation handelt. Für den Computer ist das, was wir auf dem Bildschirm sehen, lediglich eine Reihe von Bildpunkten, die je nach gewählter Bildschirmauflösung entweder schwarzweiß oder farbig sind. Die Größe des benötigten Speicherplatzes für die gesamte Bildinformation ist jedoch in jedem Fall gleich und beträgt 32000 Byte. Im hochauflösenden Modus läßt sich die Größe des Bildschirmspeichers sehr leicht erklären. Der Computer kann in dieser Auflösungsstufe 640 Bildpunkte horizontal und 400 vertikal darstellen. Das ergibt alo eine Gesamtzahl von 256000 Bildpunkten. Da jeder dieser Bildpunkte nur zwei Zustände annehmen kann, entweder weiß oder schwarz, läßt sich jeder Bildpunkt mit der kleinsten Informationseinheit des Computers, mit einem Bit, welches den Wert 0 oder 1 annehmen kann, definieren. Acht solcher Bits zusammengefaßt entsprechen einem Byte, was also bedeutet, daß man 256000 durch 8 teilen muß, um auf die Anzahl der benötigten Bytes zu kommen, womit wir bei dem Wert von 32000 Bytes angelangt wären. Um festzustellen, ab welcher Speicherstelle oder Adresse der Computer die Bildinformation im Moment abgelegt hat, bedient man sich des Befehls *XBIOS (Adresse,2)*. Hierbei wird in der Variablen *Adresse* der Wert der Speicheradresse abgelegt, ab der der Computer das momentan sichtbare Bild darstellt. Diese Adresse muß nicht immer gleich sein, sondern kann vom Programmierer an eine andere Stelle gelegt werden; auch ist es möglich, mehrere solcher Speicherbereiche anzulegen, wobei natürlich immer nur die Information eines Bereichs auf dem Schirm sichtbar ist. Das einzige, was hierbei zu beachten ist, ist, daß die Adresse eines solchen Speicherbereichs, zumindest bei einem „normalen" Atari ST, durch 256 teilbar sein muß (beim Atari STE reicht es aus, wenn sie durch 2 teilbar ist).

Mehrere Bildschirme

Natürlich kann man die Bildinformation nicht wahllos im Speicher ablegen, da man nie weiß, ob der Speicherplatz nicht schon von anderen Programmteilen besetzt ist. Aus diesem Grund ist es erforderlich, daß man sich Speicherplatz in einem RAM-Bereich reserviert, der, ohne großen Schaden anzurichten, vom Programmierer genutzt werden kann. Dazu dient der Befehl CLEAR. Wie wir wissen, benötigt ein kompletter Bildschirm 32000 Bytes. Daher muß also pro Bildschirm mindestens diese Speicherplatzmenge reserviert werden. Da ein mit CLEAR reservierter Speicherplatz jedoch nicht nur vom Programmierer allein genutzt wird, sondern auch vom sogenannten GEMDOS, einem Teil des Betriebssystems, der unter anderem für die Darstellung von Dateiauswahl- und Alarmboxen zuständig ist (OMIKRON.BASIC reserviert standardmäßig 65536 Bytes hierfür), ist es von Vorteil, großzügig zu sein und immer ein bißchen mehr zu reservieren. Wenn Speicherplatz reserviert wird, möchte man natürlich auch wissen, ab welcher Adresse man ihn nutzen kann. Das kann man mit der Funktion MEMORY in Erfahrung bringen [Beispiel: *Adresse = MEMORY (32000)*], die dann bei Angabe des zu reservierenden

Speicherplatzes die entsprechende Speicheradresse zurückgibt, ab der der Platz reserviert wurde. Wenn man auf diese Weise mehrere Bildschirme erzeugt, nützt uns das jedoch recht wenig, wenn man auf diese Schirme keine Grafik oder keinen Text ausgeben kann. Daher gibt es einen XBIOS-Befehl, der es unter anderem erlaubt, diese angelegten Speicherbereiche als sogenannte virtuelle (scheinbare) Bildschirme zu nutzen.

Sämtliche Befehle, die Text oder Grafik ausgeben, werden auf diese sozusagen im Hintergrund befindlichen, momentan also unsichtbaren, Schirme umgeleitet.

Der Befehl, der auch noch andere Funktionen beinhaltet, auf die im Rahmen dieses Artikels jedoch nicht eingegangen werden soll, sieht in diesem Fall folgendermaßen aus:

```
XBIOS (,5,HIGH(Adresse),
    LOW(Adresse),-1,-1,-1)
```

Adresse ist hierbei wieder die Adresse eines Speicherbereichs, in der die Bildinformation abgelegt werden soll.

Für das möglichst flackerfreie Sichtbarmachen dieser Bildschirme kann man sich des SCREEN-Befehls bedienen (SCREEN Schirmnummer, Adresse). Die Schirmnummer kann hierbei 0, 1 oder 2 sein, wobei 1 oder 2 vorzuziehen ist, da bei Schirmnummer 0 im sogenannten Link-Modus gearbeitet wird, das heißt, bei Textausgabe wird bei Zeilenüberschreitung ein senkrechter Strich dargestellt, der eventuell störend wirkt. Wie dieses ganze Wissen nun in der Praxis genutzt werden kann, zeigt Listing 1. In diesem Beispielprogramm werden im Hintergrund auf 10 Bildern konzentrische Kreise gezeichnet, die dann nacheinander als Computeranimation gezeigt werden.

Soft-Scrolling

In Grafikprogrammen, insbesondere bei Computerspielen, kann man oft beobachten, wie der Bildschirm zeilenweise nach oben oder unten oder auch seitwärts verschoben wird. Dieses Phänomen wird Scrolling genannt (to scroll - rollen). Wie man so eine zeilenweise Bildverschiebung in vertikaler Richtung, also von oben nach unten, oder umgekehrt vornehmen kann, soll uns nun noch ein wenig beschäftigen.

Zunächst einmal ist es sinnvoll zu überlegen, was beim Scrolling eigentlich passiert. Bei einem zeilenweisen vertikalen Scrolling wird im Grunde lediglich die Anzeigeadresse des Bildschirms um die Anzahl von Bytes verschoben, die der Bildinformation für eine Zeile entsprechen. Wieviel Bytes müßten das sein? In hoher Bildschirmauflösung entspricht jeder Bildpunkt, wie bereits bekannt, einem Bit. Bei einer Zeilenbreite von 640 Bildpunkten beträgt der benötigte Speicherplatz im Bildschirmspeicher also 640 Bits oder 80 Bytes (640/8=80). In diesem Fall müßte man also, wenn man den gesamten Bildinhalt um eine Zeile nach oben verschieben, möchte die Adresse des Bildschirmspeichers um 80 Bytes verschieben. Da jedoch die Bildschirmadresse beim „normalen" ST durch 256 teilbar sein muß, ist das nicht ohne weiteres möglich. Man kann sich hier jedoch eines kleinen Tricks bedienen. Mit Hilfe des Befehls MEMORY_MOVE ist es möglich, Speicherbereiche, die an einer durch 2 teilbaren Adresse liegen, zu kopieren. Man kopiert also den um 80 Bytes verschobenen Bildinhalt an die durch 256 teilbare Startadresse eines Bereiches, der als Bildschirmspeicher benutzt werden soll. Aber es handelt sich nicht in jedem Fall um 80 Bytes pro Zeile. In mittlerer und geringer Auflösung muß nämlich noch die Information für die verschiedenen Farben abgelegt werden. In mittlerer Auflösung hat eine Bildschirmzeile ebenfalls 640 Bildpunkte. Jeder dieser Punkte kann jedoch eine von vier verschiedenen Farben annehmen. Der Computer muß das irgendwie auseinanderhalten können.

Dies tut er, indem für jeden Bildpunkt in mittlerer Auflösung zwei Bits zuständig sind. Mit zwei Bits kann man vier verschiedene Zustände (2^2=4), oder in diesem Fall Farben, darstellen (00=1, 01=2, 10=3, 11=4). Um den Bildschirm um eine Zeile zu verschieben, muß also die Bildschirmadresse um 1280 Bits (640*2) oder 160 Bytes verschoben werden. Im Fall der geringen Auflösung, mit einer Zeilenbreite von 320 Bildpunkten und 16 Farben, müssen sich 4 Bits die Information für einen Bildpunkt teilen (2^4=16). Pro Zeile sind 320*4 Bits, also ebenfalls 160 Bytes erforderlich. Listing 2 zeigt die Anwendung dieser Erkenntnisse. Es werden insgesamt 4 Bildschirme erzeugt. Die Schirme 3 und 4 werden per Zufallsgenerator mit Text beschrieben.

Anschließend wird die Bildinformation abwechselnd in Schirm 1 und 2 jeweils um eine Zeile verschoben hineinkopiert, und zwar so, daß Schirm 1 oder 2 erst nach dem Kopiervorgang zu sehen sind. Dieses Hin- und Herschalten dient der flimmerfreien Darstellung, da bei direktem Kopieren der Bildinformation in einen bereits sichtbaren Schirm der Bildaufbau nicht ganz mit dem Programm synchron läuft. Dieser Vorgang wird solange wiederholt, bis von Schirm 3 nach Schirm 4 gescrollt wurde. Das Demoprogramm ist in allen drei Auflösungen lauffähig. Experimentieren Sie ruhig ein wenig herum. Es ist zum Beispiel leicht möglich, in Schritten von zwei Zeilen oder mehr zu scrollen. Auch braucht man den Scroll-Vorgang nicht über den ganzen Schirm auszudehnen. Wird im Listing beim MEMORY_MOVE-Befehl die Anzahl der zu kopierenden Bytes von 32000 auf 16000 beschränkt, wird beispielsweise nur in der oberen Bildschirmhälfte gescrollt. Mit ein bißchen Geschick läßt sich so eine Vielzahl von Effekten programmieren.



```
 1:   '+---------------------------------+
 2:   '|            Listing 1            |
 3:   '| DEN BILDSCHIRMSPEICHER IM GRIFF |
 4:   '|        mit OMIKRON-BASIC        |
 5:   '|            ANIMATION            |
 6:   '+---------------------------------+
 7:
 8:   CLEAR 360000:'Genügend Speicherplatz reservieren
 9:
10:   Bildschirm:'  Grundeinstellungen die mit der
                    Auflösung zu tun haben
11:   Erzeuge_Schirme 11
12:
13:   SCREEN 1,Adr_Erster_Schirm%L:' Siehe Prozedur
                                        <Erzeuge_Schirme>
14:   Cursor_Aus:'Stört nur
15:
16:   'Grafiken erzeugen
17:
18:   X%L=Maxbreite%L\2
      Y%L=Maxhoehe%L\2: 'Kreismittelpunkt
19:   FOR I%L=2 TO 11
20:
21:    'Text auf den ersten Schirm
22:
23:     Aktiviere_Schirm 1:'Schirm 1 ist gerade
                           sichtbar (SCREEN 1)
24:     LOCATE 2,1: PRINT "Bitte einen Moment Geduld"
25:     LOCATE 3,1: PRINT "Grafik Nr. ";I%L-1;" wird
                          gerade erzeugt"
26:
27:    'Grafik im Hintergrund auf die Schirme 2 bis 11
28:
29:     Aktiviere_Schirm I%L:'Zur Zeit unsichtbar im
                              Hintergrund
30:     LOCATE 2,1: PRINT "DEMO ANIMATION"
31:     FOR Radius%L=0 TO 400 STEP 20
32:       CIRCLE X%L,Y%L,Radius%L+2*I%L
33:     NEXT
34:
35:   NEXT
36:
37:   'ANIMATION                                      →
```

```
38:
39:   SCREEN 2,Adr_Erster_Schirm%L
40:   Cursor_Aus
41:   REPEAT
42:     FOR I%L=2 TO 11
43:       Schirmadresse%L=Adr_Erster_Schirm%L+(I%L-1)*
                32000
44:       WAIT 1/40:'Bildfrequenz
45:       SCREEN 2,Schirmadresse%L
46:     NEXT
47:   UNTIL INKEY$ <>"":'Abbruch bei Tastendruck
48:
49:   END
50:
51:
52:   'Standard-Prozeduren die für die Demos benötigt
      werden
53:
54:   DEF PROC Bildschirm
55:     XBIOS (Aufloesung%L,4):
                 'Bildschirmaufloesung ermitteln
56:     Maxbreite%L=640+320*(Aufloesung%L=0):
                 'Bildpunkte horizontal
57:     Maxhoehe%L=400+200*(Aufloesung%L<2):
                 'Bildpunkte vertikal
58:     CLIP 0,0,Maxbreite%L,Maxhoehe%L
59:     Bytes_Pro_Zeile%L=160+80*(Aufloesung%L=2)
60:   RETURN
61:
62:   DEF PROC Cursor_Aus
63:     PRINT CHR$(27);"f";
64:   RETURN
65:
66:   DEF PROC Erzeuge_Schirme(Nschirme%L)
67:     Adr_Erster_Schirm%L= MEMORY(32000*Nschirme%L+
           256)
68:     Adr_Erster_Schirm%L=(Adr_Erster_Schirm%L\256)*
           256+256
69:   RETURN
70:
71:   DEF PROC Aktiviere_Schirm(Nr%L)
72:     LOCAL Adr%L
73:     Adr%L=Adr_Erster_Schirm%L+(Nr%L-1)*32000
74:     XBIOS (,5, HIGH(Adr%L), LOW(Adr%L),-1,-1,-1)
75:   RETURN
```

```
 1:   '+-----------------------------------+
 2:   '|            Listing 2              |
 3:   '| DEN BILDSCHIRMSPEICHER IM GRIFF   |
 4:   '|         mit OMIKRON-BASIC         |
 5:   '|     VERTIKALES SOFT SCROLLING     |
 6:   '+-----------------------------------+
 7:   ' (c) 1991 MAXON Computer
 8:   CLEAR 150000:'Genügend Speicherplatz reservieren
 9:
10:   Bildschirm:' Grundeinstellungen die mit der
                  Auflösung zu tun haben
11:   Erzeuge_Schirme 4
12:
13:   SCREEN 1,Adr_Erster_Schirm%L:'Siehe Prozedur
                                 <Erzeuge Schirme>
14:   Cursor_Aus
15:
16:   FOR I%L=3 TO 4
17:
18:     'Text auf den ersten Schirm
19:
20:     Aktiviere_Schirm 1:'Schirm 1 ist gerade
                          sichtbar (SCREEN 1)
21:     LOCATE 2,1: PRINT "Bitte einen Moment Geduld"
22:     LOCATE 3,1: PRINT "Schirm Nr. ";I%L-1;" wird
                      gerade beschriftet"
23:
24:     'Text im Hintergrund auf die Schirme 3 und 4
25:
26:     Aktiviere_Schirm I%L:'Zur Zeit unsichtbar im
                             Hintergrund
27:     LOCATE 2,1: PRINT "DEMO VERTIKALES SCROLLING"
28:     FOR Zeile%L=4 TO 24
29:       Nspalten%L=80+40*(Aufloesung%L=0)
30:       FOR Spalte%L=0 TO Nspalten%L-1
31:         LOCATE Zeile%L,Spalte%L
32:         PRINT CHR$( RND(64)+32);:'Zeichenauswahl
                         per Zufallsgenerator
33:       NEXT
34:     NEXT
35:
36:   NEXT
37:
38:   'VERTIKALES SCROLLING
39:
40:   SCREEN 2:Cursor_Aus
41:   Schirm1%L=Adr_Erster_Schirm%L
42:   Schirm2%L=Adr_Erster_Schirm%L+32000
43:   Nbildzeilen%L=400+200*(Aufloesung%L<>2)
44:   Startadresse%L=Adr_Erster_Schirm%L+
            64000:
           'Dritter Schirm Anfang erste Zeile
45:   Zeilenoffset%L=0
46:   FOR Bildzeile%L=1 TO Nbildzeilen%L
47:     MEMORY_MOVE Startadresse%L+Zeilenoffset%L,
            32000 TO Schirm2%L
48:     SCREEN 2,Schirm2%L
49:     SWAP Schirm1%L,Schirm2%L
50:     Zeilenoffset%L=Zeilenoffset%L+
                  Bytes_Pro_Zeile%L
51:     WAIT 1/50:'Bildfrequenz
52:   NEXT
53:
54:   END
55:
56:
57:   'Standard-Prozeduren die wir für die Demos
      benötigen
58:
59:   DEF PROC Bildschirm
60:     XBIOS (Aufloesung%L,4):
                 'Bildschirmaufloesung ermitteln
61:     Maxbreite%L=640+320*(Aufloesung%L=0):
                 'Bildpunkte horizontal
62:     Maxhoehe%L=400+200*(Aufloesung%L<2):
                 'Bildpunkte vertikal
63:     CLIP 0,0,Maxbreite%L,Maxhoehe%L
64:     Bytes_Pro_Zeile%L=160+80*(Aufloesung%L=2)
65:   RETURN
66:
67:   DEF PROC Cursor_Aus
68:     PRINT CHR$(27);"f";
69:   RETURN
70:
71:   DEF PROC Erzeuge_Schirme(Nschirme%L)
72:     Adr_Erster_Schirm%L= MEMORY(32000*Nschirme%L+
           256)
73:     Adr_Erster_Schirm%L=(Adr_Erster_Schirm%L\256)*
           256+256
74:   RETURN
75:
76:   DEF PROC Aktiviere_Schirm(Nr%L)
77:     LOCAL Adr%L
78:     Adr%L=Adr_Erster_Schirm%L+(Nr%L-1)*32000
79:     XBIOS (,5, HIGH(Adr%L), LOW(Adr%L),-1,-1,-1)
80:   RETURN
```

CHECKBOXEN UNTER GEM

Uwe Hax

NEUERDINGS IST ES IN MODE GEKOMMEN, DIE GESTALTUNG VON GEM-DIALOGBOXEN AN DIE DES APPLE MACINTOSH ANZULEHNEN, WAS SICHERLICH KEINESFALLS VERKEHRT IST. WENN MAN SICH JEDOCH ANSIEHT, WAS MANCHE PROGRAMMIERER SICH DABEI FÜR EINE MÜHE GEBEN, UM ZUM BEISPIEL SOLCHE KLEINIGKEITEN WIE CHECKBOXEN UNTER GEM ZU REALISEREN, DA FRAGT MAN SICH, OB SICH DIESE LEUTE DIE FÄHIGKEITEN DES GEM ÜBERHAUPT GENAUER ANGESEHEN HABEN.

Die sogenannte Tatsache (?), daß GEM Checkboxen nicht bzw. nur unzureichend unterstützt, ist offenbar den meisten Programmierern Grund genug, die Finger davon zu lassen. Die anderen, die solche Sachen wie Checkboxen und beispielsweise runde Radiobuttons (wie vom Macintosh bekannt) benutzen wollen, machen sich andererseits einen Haufen (unnötige) Arbeit, indem sie benutzerdefinierte Objekte anlegen und das benötigte Objekt bis in die letzte Einzelheit mittels VDI-Befehlen zeichnen. Bei runden Buttons ist das ja noch einzusehen, da GEM diese von sich aus nicht anbietet, bei Checkboxen jedoch absolut nicht.

Weiß & weiß...

Checkboxen sind unter GEM normale Objekte vom Typ *G_BOX*, die mit dem Status *CROSSED* versehen sind. Sie gelten nur aus dem Grund als unbrauchbar, weil das Kreuz von GEM grundsätzlich in weißer Farbe gezeichnet wird - offensichtlich handelt es sich dabei um einen Fehler, der von Atari bis heute nicht behoben wurde. Dieses weiße Kreuz ist insofern unbrauchbar, da man für gewöhnlich einen weißen Hintergrund für Buttons benutzt - und das ist in etwa so wie die ostfriesische Nationalflagge: weißer Adler auf weißem Grund.

Nichtsdestoweniger läßt sich dieses Problem jedoch mit einem kleinen und völlig simplen Trick umgehen, wie das vorliegende Listing zeigt. Das Programm öffnet eine Dialogbox mit einer Checkbox und einem Exit-Button. Die Checkbox kann beliebig oft angeklickt werden und ändert jedesmal ihren Status von *CROSSED* auf *NORMAL* bzw. umgekehrt.

Wie funktioniert das Ganze nun? Die Antwort ist verblüffend einfach: Man nimmt eine gewöhnliche *G_BOX* und versieht sie mit dem Status *CROSSED*. Wie bereits geschildert, erhält man dadurch ein weißes und damit nicht sichtbares Kreuz auf einem weißen Hintergrund. Deshalb muß man die *G_BOX* jetzt noch mit der Hintergrundfarbe Schwarz versehen, und man bekommt ein weißes Kreuz auf einem schwarzen Hintergrund, was zwar immer noch nicht das gewünschte Ergebnis darstellt, aber schon wesentlich näher am Ziel ist; immerhin kann man jetzt schon das Kreuz vom Hintergrund unterscheiden.

Trick 17

Der eigentliche Trick kommt erst jetzt: Entweder belegt man im Resource Construction Set das Objekt mit dem Status *SELECTED* vor, oder man selektiert die Checkbox im Programm vor der Benutzung einmal; dies wird auch so im Beispielprogramm gemacht. Um zu verstehen, warum man dadurch plötzlich eine „richtige" Checkbox erhält, muß man wissen, wie GEM den Zustand *SELECTED* bearbeitet: Es wird nämlich das gesamte Objekt, in diesem Fall eine *G_BOX* mit dem Status *CROSSED* und einem schwarzen Hintergrund, komplett invertiert. Dadurch erhält man dann nichts anderes als eine *G_BOX* mit einem weißen Hintergrund und einem schwarzen Kreuz - also eine „richtige" Checkbox!

Wenn der Button angeklickt wird, ist nun nichts anderes mehr zu tun, als den Zustand *CROSSED* zu löschen bzw. bei erneutem Anklicken wieder zu setzen und anschließend die Checkbox neu zu zeichnen. - Voilà!

Radiobuttons

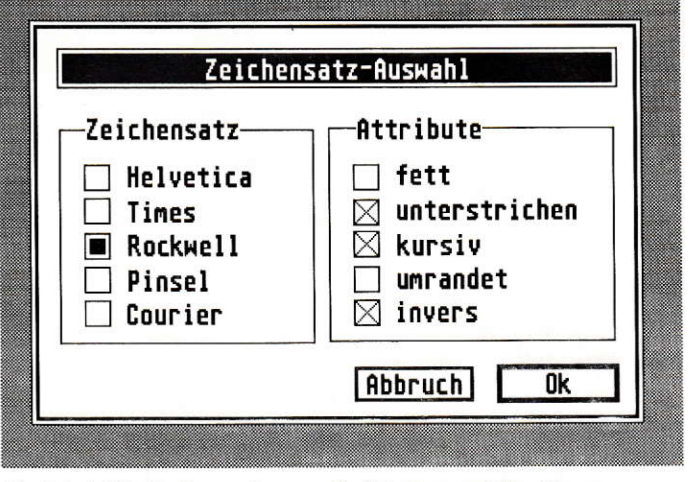
Abschließend noch ein Wort zum Thema Radiobuttons. Wie oben bereits erwähnt, benutzen viele Programme mittlerweile die Macintosh-typischen runden Radiobuttons, die sicherlich auch optisch eindrucksvoll sind. Wer sich jedoch nicht soviel Mühe machen will, dem steht auch hier unter GEM eine wesentlich einfachere Möglichkeit offen, die sich meiner Meinung nach optisch auch besser ins GEM eingliedert, da runde Objekte in GEM für gewöhnlich nicht vertreten sind.

Man nimmt dazu einfach wieder eine *G_BOX* und versieht sie mit dem Status *OUTLINED*. Bei Anklicken muß die Box nur noch selektiert werden, was GEM ja bekanntlich automatisch macht, und schon hat man einen Radiobutton, der zwar eckig statt rund ist, sich aber ansonsten weder in Aussehen

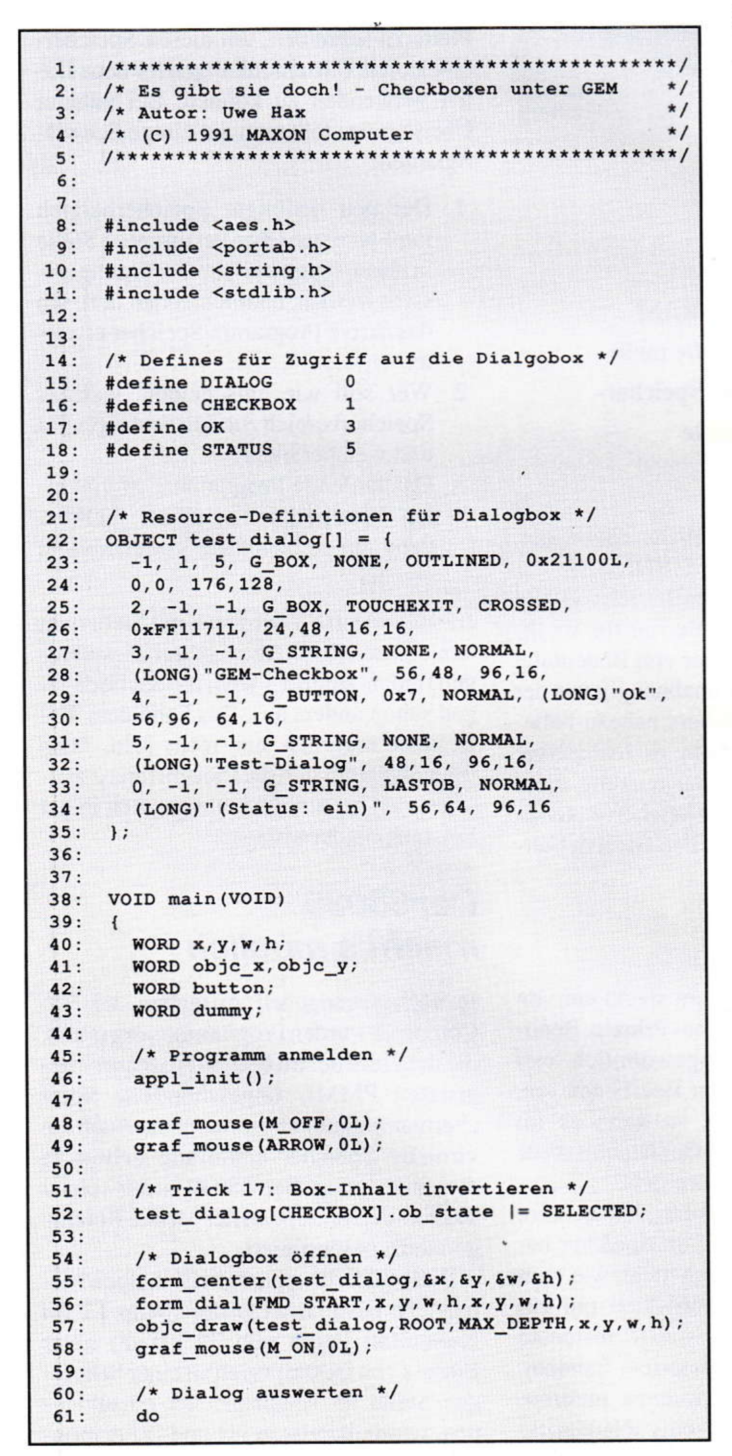
noch in Funktionsweise von den runden Radiobuttons unterscheidet. Noch dazu hebt er sich von den oben eingeführten Checkboxen sehr gut ab und ist lange nicht so aufwendig zu programmieren wie ein runder Radiobutton, für den wieder einmal ein benutzerdefiniertes Objekt angelegt werden muß. Die Abbildung zeigt, wie eine solche Dialogbox aussehen könnte; die Unterschiede zwischen den beiden Button-Typen sind deutlich zu erkennen.

Mit anderen Worten: Auch mit „normaler“ GEM-Programmierung läßt sich immer noch sehr viel mehr aus GEM herausholen, als den meisten Leuten bisher bekannt zu sein





Ein Beispiel für die Verwendung von Radiobuttons und Checkboxen



```
  1: /************************************************/
  2: /* Es gibt sie doch! - Checkboxen unter GEM    */
  3: /* Autor: Uwe Hax                              */
  4: /* (C) 1991 MAXON Computer                     */
  5: /************************************************/
  6:
  7:
  8: #include <aes.h>
  9: #include <portab.h>
 10: #include <string.h>
 11: #include <stdlib.h>
 12:
 13:
 14: /* Defines für Zugriff auf die Dialgobox */
 15: #define DIALOG      0
 16: #define CHECKBOX    1
 17: #define OK          3
 18: #define STATUS      5
 19:
 20:
 21: /* Resource-Definitionen für Dialogbox */
 22: OBJECT test_dialog[] = {
 23:   -1, 1, 5, G_BOX, NONE, OUTLINED, 0x21100L,
 24:   0,0, 176,128,
 25:   2, -1, -1, G_BOX, TOUCHEXIT, CROSSED,
 26:   0xFF1171L, 24,48, 16,16,
 27:   3, -1, -1, G_STRING, NONE, NORMAL,
 28:   (LONG)"GEM-Checkbox", 56,48, 96,16,
 29:   4, -1, -1, G_BUTTON, 0x7, NORMAL, (LONG)"Ok",
 30:   56,96, 64,16,
 31:   5, -1, -1, G_STRING, NONE, NORMAL,
 32:   (LONG)"Test-Dialog", 48,16, 96,16,
 33:   0, -1, -1, G_STRING, LASTOB, NORMAL,
 34:   (LONG)"(Status: ein)", 56,64, 96,16
 35: };
 36:
 37:
 38: VOID main(VOID)
 39: {
 40:   WORD x,y,w,h;
 41:   WORD objc_x,objc_y;
 42:   WORD button;
 43:   WORD dummy;
 44:
 45:   /* Programm anmelden */
 46:   appl_init();
 47:
 48:   graf_mouse(M_OFF,0L);
 49:   graf_mouse(ARROW,0L);
 50:
 51:   /* Trick 17: Box-Inhalt invertieren */
 52:   test_dialog[CHECKBOX].ob_state |= SELECTED;
 53:
 54:   /* Dialogbox öffnen */
 55:   form_center(test_dialog,&x,&y,&w,&h);
 56:   form_dial(FMD_START,x,y,w,h,x,y,w,h);
 57:   objc_draw(test_dialog,ROOT,MAX_DEPTH,x,y,w,h);
 58:   graf_mouse(M_ON,0L);
 59:
 60:   /* Dialog auswerten */
 61:   do
```

```
 62:   {
 63:     button=form_do(test_dialog,0) & 0x7fff;
 64:
 65:     /* Checkbox angeklickt? */
 66:     if (button==CHECKBOX)
 67:     {
 68:       /* Kreuz ein- und ausschalten */
 69:       if (test_dialog[CHECKBOX].ob_state &
 70:                                        CROSSED)
 71:       {
 72:         test_dialog[CHECKBOX].ob_state &=
 73:                                       ~CROSSED;
 74:         strncpy(&test_dialog[STATUS].ob_spec.
 75:                         free_string[9],"aus",3);
 76:       }
 77:       else
 78:       {
 79:         test_dialog[CHECKBOX].ob_state |=
 80:                                        CROSSED;
 81:         strncpy(&test_dialog[STATUS].ob_spec.
 82:                         free_string[9],"ein",3);
 83:       }
 84:
 85:       /* Checkbox neu zeichnen */
 86:       graf_mouse(M_OFF,0L);
 87:       objc_offset(test_dialog,CHECKBOX,&objc_x,
 88:                                       &objc_y);
 89:       objc_draw(test_dialog,ROOT,MAX_DEPTH,
 90:                 objc_x,objc_y,
 91:                 test_dialog[CHECKBOX].ob_width,
 92:                 test_dialog[CHECKBOX].ob_height);
 93:       graf_mouse(M_ON,0L);
 94:
 95:       /* Status neu zeichnen */
 96:       objc_offset(test_dialog,STATUS,&objc_x,
 97:                   &objc_y);
 98:       objc_draw(test_dialog,ROOT,MAX_DEPTH,
 99:                 objc_x,objc_y,
100:                 test_dialog[STATUS].ob_width,
101:                 test_dialog[STATUS].ob_height);
102:
103:       /* Warten, bis Mausclick-Ende */
104:       evnt_button(1,1,0,&dummy,&dummy,&dummy,
105:                   &dummy);
106:     }
107:   }
108:   while (test_dialog[button].ob_flags &
109:                                    TOUCHEXIT);
110:
111:   /* Dialogbox entfernen und Programm beenden */
112:   form_dial(FMD_FINISH,x,y,w,h,x,y,w,h);
113:   appl_exit();
114:   exit(0);
115: }
```

Virtuelle Speicherverwaltung



Eine Fallstudie

Auch wenn der TT standardmäßig nur mit 4 MByte TT-RAM ausgeliefert wird, lassen sich Anwendungen realisieren, die mehr als 4 MByte Speicherplatz benötigen. Dies ist auch ohne Speichererweiterung denkbar, denn der TT erlaubt die sogenannte „virtuelle Speicherverwaltung“.

Die Tendenz ist eindeutig: Mit wachsender Größe des Hauptspeichers wächst auch der Speicherbedarf von Programmen. Galt es früher noch als utopisch, mehr als 640 kByte RAM zu benötigen, so stellt dieser Wert heute eher eine Untergrenze dar. Diejenigen, die bereits mit MS-DOS Bekanntschaft gemacht haben, können ein Lied davon singen.

Der Atari ST wird schon seit langem mit mindestens einem MByte Speicher ausgeliefert. Die neuen Rechnergenerationen von Atari, unter denen der MegaSTE und der TT zu verstehen sein sollen, bieten standardmäßig bereits zwei bzw. vier Megabyte Hauptspeicher. Dies sollte doch eigentlich für die meisten Anwendungen ausreichen, oder?

Speicherhunger

Nun, man sollte diese Frage nicht einfach bejahen. So spricht die Entwicklung auf dem ST-Sektor eine deutliche Sprache. Waren für den ST bis vor kurzem 4 MByte RAM das Maximum, so sind inzwischen Speichererweiterungen auf dem Markt, die eine Aufrüstung auf bis zu 12 MByte ermöglichen. Die Preise für solche Lösungen bewegen sich momentan jedoch noch jenseits von Gut und Böse. Der TT kann alles in allem auf bis zu 26 MByte aufgerüstet werden. Unter TOS ist dieser Speicherausbau sicherlich nicht notwendig, aber da ist ja auch noch UNIX ...

Apropos UNIX: Dieses Betriebssystem, das in Zukunft nicht mehr nur für Workstations und Großrechner von Bedeutung sein dürfte, erlaubt es, unabhängig von der Größe des Hauptspeichers nahezu beliebig große Anwendungen zu realisieren. Dies geschieht durch Ausnutzung einer speziellen Technik zur Speicherorganisation, nämlich der virtuellen Speicherverwaltung.

Aufklärung

Hinter diesem Schlagwort steckt ein von der Idee her recht einfaches Prinzip. Benötigt ein Programm ungewöhnlich viel Speicherplatz (z.B. zum Bearbeiten von gescannten Vorlagen), so kann es im Hauptspeicher auch bei Geräten mit mehr als 4 MByte RAM eng werden.

Ähnlich sieht es aus, wenn sich mehrere Programme gleichzeitig im Speicher befinden. Dabei muß es sich nicht, wie vom ST her bekannt, ausschließlich um ein Hauptprogramm sowie einige residente Programme (z.B. Accessories) handeln, sondern es können durchaus mehrere Hauptprogramme gleichzeitig ablaufen. So erlaubt es das Programm MULTIGEM, verschiedene Programme auf ST oder TT quasi nebeneinander laufen zu lassen.

Wird nun mehr Speicherplatz benötigt, als das RAM hergibt, werden durch die virtuelle Verwaltung Speicherbereiche, die momentan nicht benötigt werden, auf die Festplatte ausgelagert. So wird Speicherplatz frei, der anschließend für neue Daten verwendet werden kann. Die Festplatte wird also als Speichererweiterung verwendet.

Leicht gesagt

Der eine oder andere Leser dürfte nun ins Grübeln geraten. Schließlich nützt es nichts, einen Datenbereich einfach auf die Platte zu schreiben, um diesen Speicherblock ohne Einschränkungen für neue Daten verwenden zu können. Bei näherer Überlegung stößt man auf einige Schwierigkeiten:

1. Der neu benötigte Speicherbereich muß an einer ganz bestimmten Stelle im Hauptspeicher zur Verfügung gestellt werden, nämlich genau dort, wo das aktive Programm Speicher erwartet.
2. Wer soll wie entscheiden, welcher Speicherbereich zur Zeit benötigt wird und welcher nicht?
3. Das laufende Programm darf von allen Manipulationen nichts merken, denn nur so bleibt die Kompatibilität gewahrt.

Auf dem Atari ST wird man mit Sicherheit keine brauchbare Antwort auf diese Fragen finden. Beim TT sieht die Sache dagegen schon anders aus. Das kann dem TT-Besitzer natürlich nur recht sein, denn irgendwo muß sich die Preisdifferenz zwischen den beiden Rechnern ja auch in der Leistung niederschlagen.

Der 68030 macht's möglich

In vorangegangenen Ausgaben der ST-Computer wurden Programme vorgestellt, die mit Hilfe der im 68030-Prozessor integrierten PMMU ungewöhnliche Speichermanipulationen vornehmen. Auch die virtuelle Speicherverwaltung gehört in diesen Rahmen. Der 68030 ist für solche Tricks aufgrund spezieller MMU-Befehle geradezu prädestiniert.

Daß es überhaupt möglich ist, Speicherbereiche bestimmter Größe (beim TT im Normalfall Pages von 32 kByte) unter Einsatz von Deskriptoren an einer beliebigen Stelle im Hauptspeicher einzublenden, wurde bereits in [1] und [2] demonstriert. Somit kann Speicherplatz an genau der Stelle zur Verfügung gestellt werden, an der neuer Speicher benötigt wird. Hierzu genügt eine Manipulation von Seiten-

deskriptoren. Wie erfährt man aber nun, wann an welcher Stelle im Adreßraum Speicherplatz zur Verfügung gestellt werden muß? Durch das Auftreten eines Busfehlers!

Busfehler erwünscht

Eigentlich mag es der Atari-Anwender gar nicht, wenn sich während der Arbeit ein solcher Fehler in Form zweier Bomben Luft macht. Systemabstürze sind halt stets unangenehm. Aber gerade die Tatsache, daß auch der Zugriff auf nicht vorhandenes RAM oder einen ungültigen Deskriptor zu einer Exception in Form eines Busfehlers führt, ist für die virtuelleSpeicherverwaltung von Bedeutung. Ein speicherresidentes Programm kann auf diesen Fehlerzustand reagieren und den Fehler möglicherweise beheben. Wird also ein Busfehler dadurch hervorgerufen, daß Software auf einen RAM-Bereich zugreift, der momentan nicht im Speicher vorhanden ist, kann ein geeignetes Programm den Fehler analysieren und den benötigen Speicherblock zur Verfügung stellen. Dafür muß natürlich ein anderer Speicherbereich geopfert, also auf die Festplatte geschrieben werden. Wird nun im weiteren Programmverlauf auf eben diesen Bereich zugegriffen, wird nach dem gleichen Schema verfahren. Es wird also wieder ein Bereich des TT-RAMs auf die Platte geschrieben, und der ursprünglich ausgelagerte Block wird an alter Stelle wieder in den Speicher geholt.

Busfehler != Busfehler

Für die virtuelle Verwaltung des TT-RAMs kann ein Busfehler also ein Zeichen dafür sein, daß ein Speicherblock benötigt wird, der sich momentan nicht im Hauptspeicher befindet. Allerdings ist nicht auszuschließen, daß die Exception durch einen Fehlerzustand hervorgerufen wurde, der mit der für uns interessanten Speicherverwaltung überhaupt nichts zu tun hat.

Zunächst geht es also darum, die Ursache des Busfehlers näher zu analysieren. Dabei ist eine bestimmte Fehlerquelle von besonderer Bedeutung: ein ungültiger Seitendeskriptor.

Wie bereits in den vorausgegangenen Ausgaben der ST-Computer erläutert, ist ein Seitendeskriptor Bestandteil einer Tabelle, die definiert, wie der logische dem physikalischen Speicher zugeordnet werden soll. Da es verschiedene Arten von Deskriptoren gibt (Seiten- und Tabellendeskriptoren), werden diese anhand der niederwertigen beiden Bits des Deskriptors unterschieden. Sind beide Bits 0, kann die MMU keine Zuordnung vornehmen, es

B	L	S	0	W	I	M	0	0	T	0	0	0	Levels

PMMU-Statusregister

liegt ein ungültiger Deskriptor vor. Anders ausgedrückt: Es gibt keinen Hinweis darauf, wie der logische Speicher, der durch diesen Deskriptor beschrieben wird, auf den physikalischen Speicher abgebildet werden soll.

Das Vorliegen eines ungültigen Deskriptors kann nun unter Programmkontrolle von anderen Ursachen für einen Busfehler unterschieden werden. Es läßt sich nämlich feststellen, ob der Zugriff auf eine bestimmte Adresse zu einem gültigen oder ungültigen Deskriptor führt. Ist der Deskriptor gültig, handelt es sich in der Tat um einen echten Busfehler. Ein ungültiger Deskriptor dagegen kann dazu verwendet werden, eine virtuell verwaltete Speicherseite zu markieren, die sich momentan nicht im Hauptspeicher befindet.

Ein wichtiger Test

Um die Ursache eines Busfehlers zu ermitteln, muß die Gültigkeit der Adresse, die den Fehler hervorgerufen hat, überprüft werden. Diese Aufgabe erfüllen MMU-Befehle des Typs PTEST. PTESTR überprüft die Möglichkeit, einen Lesezugriff auf eine bestimmte Adresse durchzuführen, PTESTW kümmert sich um Schreibzugriffe. Die Syntax für diese Befehle ist recht ungewöhnlich, da bis zu vier Parameter möglich sind. Allgemein haben die PTEST-Befehle die folgende Form:

```
PTEST <fc>,<ea>,#level[,An]
```

fc stellt den Zustand der Function Code Bits FC0-FC3 dar, der für den Test zugrunde gelegt werden soll. Diese Bits erlauben nähere Aussagen über einen Buszugriff, so z.B., ob dieser aus dem Supervisor- oder User-Modus heraus erfolgte. Nähere Informationen finden sich in der Fachliteratur [3].

ea enthält die Adresse, deren Zulässigkeit überprüft werden soll. Nach einem Busfehler befindet sich diese Adresse mit dem Offset 16 auf dem Interrupt-Stack des 68030.

level gibt an, bis zu welcher Ebene die Deskriptortabellen durchsucht werden sollen, um die Gültigkeit der Adresse ea festzustellen. Eine 7 sorgt dafür, daß der zugehörige Seitendeskriptor bei Bedarf bis zur letzten Tabllenebene gesucht wird. Bei einer 0 werden nur diejenigen Deskriptoren berücksichtigt, die sich im Adress Translation Cache (ATC) der PMMU befinden. (Dies sind beim 68030 bis zu 22, bei der PMMU 68851 bis zu 64 Einträge.)

Die Angabe eines Adreßregisters *An* ist schließlich optional. Wird ein Register spezifiziert, wird die Adresse des Dekriptors, der für die getestete Adresse zuständig ist, in diesem Adreßregister abgelegt.

Per *PTEST* kann überprüft werden, ob ein Speicherzugriff auf eine gültige Adresse führt. Nach der Befehlsausführung werden zudem einige Bits im MMU-Statusregister gesetzt.

Das MMU-Statusregister

Neben dem Prozessor-Statusregister SR, das Bestandteil aller Prozessoren der 68000-Familie ist, findet sich das PSR (auch MMUSR genannt) nur bei Prozessoren mit eingebauter PMMU (also 68030 und 68040) sowie natürlich bei den externen MMUs. Der Befehl PTEST erlaubt es, diverse Bits in diesem Register gemäß der Gültigkeit einer Adresse zu setzen oder zurückzusetzen:

B (BUSERR): Dieses Bit zeigt an, daß die getestete Adresse einen Busfehler verursachte, als deren Deskriptor gesucht wurde.

L (Limit Violation): Liegt eine Limit-Überschreitung bei einen Deskriptor im Langformat vor, wird dieses Bit gesetzt.

S (Supervisor Only): Es ist möglich, Speicherseiten gegen einen Zugriff aus dem User-Modus zu schützen. Wird versucht, eine solche Seite dennoch im User-Modus anzusprechen, wird das S-Bit aktiviert.

W (Write Protected): Das W-Bit besagt, daß auf eine Adresse, die nur zum Lesen freigegeben ist, ein Schreibzugriff erfolgen sollte.

I (Invalid): Dieses Bit ist für die virtuelle Verwaltung des Speichers von besonderer Bedeutung. Es zeigt an, daß die PMMU bei der Adreßberechnung auf einen ungültigen Deskriptor gestoßen ist.

werden. Somit kann die Busfehler-Exception beendet werden. Hierzu dient der RTE-Befehl, der den 68030 dazu veranlaßt, den fehlerhaften Buszyklus zu wiederholen. Diese Möglichkeit sieht der 68000 des Atari ST übrigens nicht vor. Zwar existiert auch bei diesem Prozessor das RTE-Kommando, eine erneute Ausführung von Befehlen ist jedoch nicht vorgesehen.

Hinweise für Programmierer

Virtuelle Speicherverwaltung sollte auf dem TT ohne größere Kompatibilitätsprobleme möglich sein. Fehler entstehen lediglich dann, wenn ein Programm Interrupt-Vektoren, die periodisch aufgerufen werden, ins TT-RAM legt. Wird eine Speicherseite, auf die ein solcher Vektor zeigt, ausgelagert, so führt dies beim nächsten Interrupt zu Problemen. Häufig ist es dann nicht schnell genug möglich, die benötigten Daten nachzuladen. Dies wird mit einem Absturz quittiert. Ein Programm, das virtuelle Speicherverwaltung ermöglicht, sollte es deshalb erlauben, einzelne Speicherseiten gegen das Auslagern auf Platte zu schützen. In diesen Speicherseiten können dann Interrupt-Routinen oder resetfeste Programme installiert werden. Um keine Inkompatibilitäten zu provozieren, sollten Programme, die eine eigene Busfehler-Behandlung besitzen, auf eine virtuelle Verwaltung des TT-RAMs vorbereitet sein. Tritt ein Busfehler auf, muß geprüft werden, ob der Fehler tatsächlich durch nicht vorhandenes RAM oder nur durch einen ungültigen Deskriptor entstanden ist. Im letzten Fall darf der Programmablauf nämlich auf keinen Fall abgebrochen werden. Stattdessen muß der alte Busfehler-Vektor angesprungen werden. Falls die Exception nicht durch einen ungültigen Deskriptor hervorgerufen wurde, liegt ein Fehler vor, der nicht von einer virtuellen Speicherverwaltung aufgefangen werden kann. Nur in diesem Fall sind eigene Routinen zur Fehleranalyse erlaubt.

Also, Programmierer aufgepaßt! Um solchen Schwierigkeiten, die eine virtuelle Verwaltung des TT-RAM erschweren, aus dem Weg zu gehen, schlage ich eine Routine gemäß dem abgedrucken Assembler-Listing vor. Anhand des MMU-Statusregisters wird hier zunächst die Ursache der Busfehler-Exception überprüft. Handelt es sich um einen für die virtuelle Speicherverwaltung angelegten ungültigen Deskriptor, so muß die ursprüngliche Busfehler-Routine angesprungen werden. Nur wenn der Fehler anderweitig verursacht wurde, dürfen eigene Routinen in Aktion treten.

Auch dann, wenn Sie mit einem ST arbeiten, sollten Sie die obigen Hinweise beherzigen, die TT-Anwender werden es Ihnen danken.

Man beachte ...

Abschließend noch ein Nachtrag zu dem in Heft 4 vorgestellten Patch für den TEMPUS-Editor [5]. Herrn G. Bruhn verdanke ich den Hinweis, daß der Absturz auf dem TT beim Verlassen von TEMPUS möglicherweise gar keiner ist. Beim TT ist die externe Synchronisation (in der Vergangenheit des öfteren als Bildschirmschoner mißbraucht) dann eingeschaltet, wenn Bit 0 des Video-Registers $FF820A gelöscht ist. Beim ST besagt dieser Zustand genau das Gegenteil. Atari hat also die Bedeutung dieses Bits geändert. Dieser Umstand dürfte ein mehr als ausreichendes Argument dafür sein, zu anderen Methoden beim Dunkelschalten des Bildschirms zu greifen. Empfehlenswert ist das Invertieren des Bildes bei Monochrommonitoren oder das Ändern der Farbpalette bei Farbmonitoren.

US

Literatur:

[1] „TT-Tuning - Speed Without The Price", ST-Computer 3/91
[2] „Speichermanipulationen", ST-Computer 5/91
[3] Steve Williams, „68030 Assembly Language Reference", Addison-Wesley Publishing Company Inc.
[4] Script zur Vorlesung „Betriebssysteme", Uni Kaiserslautern
[5] „TT-Manipulation auf 24 Bit", ST-Computer 4/91



Objektorientierte Programmierung mit C

Jeder, der schon mal das Vergnügen hatte, größere Programme zu schreiben, wird sicher zugeben, daß sich gewisse Strukturen, seien es Algorithmen oder Datenverbände, besonders dazu eignen, viele der anfallenden Probleme mehr oder weniger elegant zu lösen, und daß sich diese ständig nahezu unverändert in seinen Werken wiederholen. Leider nur nahezu. Die kleinen Unterschiede geben dem Speicherplatz- und Rechenzeitbewußten genug Anlaß, jedesmal eine neue Implementation vorzunehmen. Die Software-Krise ist das Resultat dieser Denkweise.

Diese Tatsache ist uns schon seit langem bekannt und betrifft vor allem Software-Hersteller, die ihre Programme für jeden Kunden neu entwerfen. Modularisierung war der erste Ansatz zur Lösung. Ohne sie würde es wohl keine anwendbare Software mehr geben. Firmen, die sich nur mit der Herstellung solcher Module (im allgemeinen Funktionsbibliotheken) beschäftigen, sind aufgetaucht und entlasten Programmierer von ständig wiederkehrenden Aufgaben. Probleme tauchen aber wieder auf, wenn man Teile dieser Module ändern möchte. Ohne Quelltexte ist dies unmöglich, mit ihnen teuer und langwierig; wer möchte sich schon seitenweise mit fremdem Programm-Code auseinandersetzen? Langer Rede kurzer Sinn: eine universelle Lösung muß her! Eine Lösung, die es uns erlaubt, vorgefertigte Software-Bausteine zu verwenden und zu erweitern, ohne den Quelltext zu besitzen.

OOP

Die objektorientierte Programmierung (OOP) scheint es zu sein. Sie führt zusätzlich zu den uns schon bekannten Sprachkonstrukten das Objekt ein. In konventionellen, also strukturierten Programmiersprachen sind Daten und Algorithmen lose im Quelltext miteinander verbunden. Ein Objekt hingegen ist eine Sammlung von Daten und den Routinen, die mit diesen Daten umgehen. Da dieser Artikel vor allem den C-Freunden gilt, werde ich in diesem zusammenfassenden Kapitel eine C-ähnliche Syntax verwenden, die so zwar nicht in C++ oder Objective-C vorkommt, die aber die Betrachtung wichtiger Bestandteile der OOP erleichtert. Ein Objekt ist dann also eine C-Struktur, in der zusätzlich zu den Variablen auch noch Funktionen, die mit diesen Variablen umgehen, definiert werden. Diese werden mit derselben simplen Syntax aufgerufen, wie man einzelne Variablen einer Struktur anspricht, zum Beispiel: *fenster.zeichnen*. Diese Vereinigung der Daten und der dazugehörigen Algorithmen ist die einzige Gemeinsamkeit unter den objektorientierten Programmiersprachen, die inzwischen von Ada bis Smalltalk reichen und mehr oder weniger objektorientiert sind. Ada bietet das Minimum: dadurch, daß nur Funktionen und Prozeduren, die im Rumpf eines Objekts definiert wurden, auf dessen Variablen zugreifen dürfen, hat man den Vorteil der Verkapselung, einer Art galvanischen Trennung, durch die unsaubere Programmierung verhindert wird. Ein gewichtiger Grund für das amerikanische Verteidigungsministerium; uns jedoch interessieren weitere Features zum Beispiel von Smalltalk, der objektorientierten Programmiersprache schlechthin. Und da wäre an erster Stelle die Vererbung zu nennen. Es ist Ihnen sicher schon aufgefallen, daß sich viele Strukturen aus der realen Welt nur in feinsten Details unterscheiden: eine Chromkassette unterscheidet sich von einer Metallkassette nur in der Beschaffenheit des Bandes; das Gehäuse kann völlig identisch sein. Auch die Datenstrukturen ähneln sich: ein Fenster hat mit einer Dialogbox zum Beispiel die Koordinaten gemeinsam (Variablen) und die Notwendigkeit den Hintergrund zu sichern (Funktionen). Man müßte nun ein Objekt definieren, das die gemeinsamen Eigenschaften beider Strukturen besitzt. In unserem Fall wäre dies also ein Rechteck, das, bevor es gezeichnet wird, den Hintergrund sichert. Die Methode, so werden Funktionen in OOP genannt, die dies tut, nennen wir für unser Beispiel *rechteck.retten*. Nun definieren wir zwei unterschiedliche Nachkommen dieses Objekts, die automatisch alle Eigenschaften erben, und ergänzen diese entsprechend. Sehen Sie dazu die Abbildung 1. Ohne daß in dem Objekt *fenster* die Methode *retten* definiert oder deklariert wurde, kann sie aufgerufen werden: *fenster.retten*. Wenn vererbte Methoden nicht die gewünschte Funktion erfüllen, können sie überladen, ergänzt oder gelöscht werden. Durch die Vererbung wird dem Programm eine hierarchische Struktur aufgesetzt. Wenn wir eine grafische Benutzeroberfläche als Beispiel nehmen, könnte diese Hierarchie wie in Abbildung 2 dargestellt aussehen.

In strukturierten Programmiersprachen wird meistens frühes Binden angewandt, d.h. daß ein Aufruf einer Funktion und die Funktion selbst bereits während der Übersetzung miteinander verbunden werden. In puren objektorientierten Sprachen wird jedoch das späte Binden angewandt. Es ist dadurch möglich, *objekt.retten* zu schrei-

ben, ohne daß der Compiler weiß, welches Objekt eigentlich gemeint ist (*fenster, rechteck, dialogbox...*). Erst wenn der Platzhalter *objekt* durch das tatsächliche Objekt ersetzt wird, also während der Laufzeit, weiß das Programm, wohin es denn nun springen muß.

Das waren die wichtigsten objektorientierten Features. Wir werden uns nun spezialisieren und nicht mehr OOP allgemein betrachten, sondern die Hybridsprachen. Eine objektorientierte Hybridsprache ist ein Amalgam aus einer konventionellen Programmiersprache und objektorientierten Features. So wurde zum Beispiel Pascal zu Object Pascal erweitert. Wir werden aber C++ und Objective-C betrachten, da C von den meisten Programmierern eingesetzt wird und beide Programmiersprachen für den ST verfügbar sind. Ich gehe in dem Artikel also davon aus, daß Sie ANSI-C gut beherrschen. Da ich viele spezielle Begriffe verwenden muß, sollte es an der Zeit sein, diese kurz zu erläutern. Der Unterschied einer Deklaration zu einer Definition, um den es hier oft geht, wird in der Literatur nicht immer sehr gut erklärt. Eine Deklaration sagt dem Compiler lediglich, welchen Typs eine Variable ist, die bereits anderswo definiert wurde oder erst später definiert wird (Vorausdeklaration). Sie können diese Variable nun verwenden, und der Compiler wird wissen, daß er nach der Definition suchen muß. Diese sagt dem Übersetzer, daß er tatsächlich eine Variable eines Typs anlegen soll. Das selbe gilt für den Unterschied zwischen einer Klasse und einem Objekt: die Klasse ist die Deklaration, wie ein Objekt auszusehen hat, und das Objekt ist eine Instanz, eine Definition also. Instanzierung ist der Vorgang, bei dem aus einer Klasse ein Objekt geschaffen wird. Auch Begriffe wie Routine, Funktion und Algorithmus bergen kleine Unterschiede, werden hier aber als äquivalent angesehen. Eine Methode ist eine Funktion innerhalb eines Objekts. Mit Methoden sendet man Nachrichten an das Objekt. Das sollte Sie aber nicht verwirren: eine Nachricht ist im Grunde genommen nichts als ein Funktionsaufruf. Noch ein Wort zu den Listings und Abbildungen. Diese sind soweit wie möglich aufeinander abgestimmt und erfüllen keine besondere Funktion, außer daß sie die Erläuterungen im Text bildlich unterstreichen.

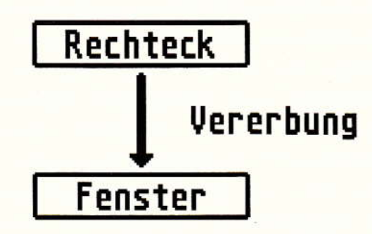


Abb. 1: Die automatische Vererbung aller Eigenschaften

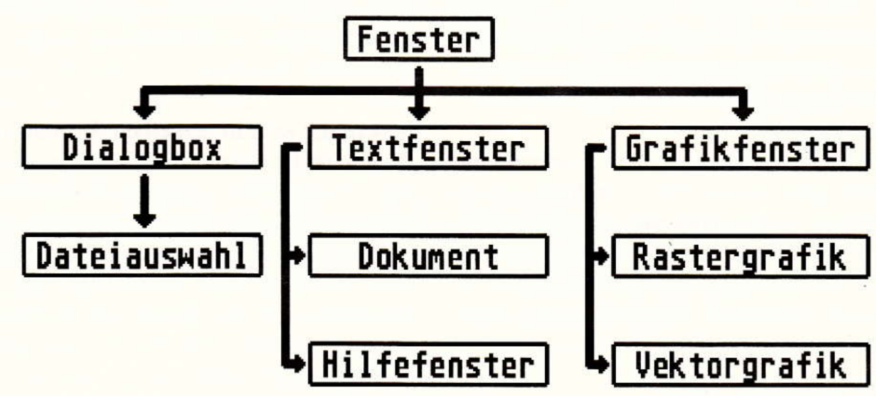


Abb. 2: Objekthierarchie

C++

Wie C wurde auch C++ an den Bell-Laboratorien entwickelt. Ziel des Projekts war, die bereits hinreichend bekannten Schwächen von C zu eliminieren und objektorientierte Zusätze mit der Sprache zu verschmelzen. Der Sprachumfang wuchs dadurch beträchtlich und brachte neue Schwierigkeiten. So blieb es nicht bei der Version 1.2. Heute unterstützen alle wichtigen Compiler auf dem PC C++ 2.0. Die allerneueste Version 2.1 brachte keine wichtigen Neuigkeiten mehr; dadurch daß C++ aber immer wichtiger wird, sah sich ANSI veranlaßt, die Sprache zu standardisieren. Der Entwurf des Standards ist noch immer im Gange.

Was ist neu? Wir werden uns zuerst die Erweiterungen anschauen, die eigentlich nichts mit OOP gemeinsam haben. Zuerst wurde C++ an das ANSI-C angepaßt beziehungsweise umgekehrt. Und wie es sich für eine neue Sprache auch gehört, wurde eine neue Art der Kommentare eingeführt, schließlich sollte man das Neue bereits an der ersten Zeile erkennen können. Sie werden mit "//" eingeführt und enden mit dem Ende einer Quelltextzeile. Mega in.

Weiterhin wurde *call by reference* implementiert, um Funktionen, die mehr als einen Parameter zurückgeben, zu vereinfachen. In C hat man das allgemein so erledigt:

```
funktion(&parameter1, &parameter2);
```

Der Funktion werden also Zeiger auf die Variablen übergeben. Um an die eigentlichen Werte zu gelangen, müssen die Zeiger erst dereferenziert werden:

```
int funktion(int *par1_pointer, ...)
  { *par1_pointer += 10; }
```

In C++ geht das einfacher, indem man den Referenzoperator verwendet:

```
int funktion(int& parameter1, ...)
  { parameter1 += 10; }
```

Der Aufruf dieser Funktion sieht dann so aus:

```
funktion(parameter1, parameter2);
```

Es werden bei der Übergabe nicht Kopien der Variablen angelegt, sondern Zeiger übergeben, die aber automatisch dereferenziert werden. Weiterhin ist es bei der Parameterübergabe möglich, sogenannte Default-Parameter zu setzen. Nehmen wir an, Sie hätten eine Funktion, die ein Fenster öffnet. Wird beim Aufruf ein Name übergeben, soll das Fenster genau diesen bekommen, *Neu* sonst.

```
int open_window(char *name = „Neu“) { ... }
```

Nun sind beide Aufrufe möglich und auch legal:

```
open_window(„complex.h“); open_window();
```

Eine Form der manuellen Optimierung in C++ ist die Möglichkeit, Funktionen als *inline* zu deklarieren. Der Compiler ersetzt dann jeden Aufruf mit der Funktion selbst. Das verbraucht zwar Speicherplatz, macht aber die gesamte Parameterübergabe über den Stack überflüssig. Einfache Funktionen, die oft aus einer Schleife heraus aufgerufen werden, sollten *inline* sein, weil dadurch die Ausführungszeit deutlich verkürzt wird.

Das nächste Feature hat schon mehr mit OOP gemeinsam. Das Überladen (overloading) wird angewandt, damit Funktionen (oder Operatoren!) gleichen Namens, die mit unterschiedlichen Parametern arbeiten, mehrmals definiert werden können:

```
int laden(BILD *raster);
  int laden(TEXT *dokument);
```

In der Version 1.2 war es noch notwendig, globale Funktionen, die überladen werden sollten, zu deklarieren; das ist jetzt nicht mehr so. Sehr interessant ist das Überladen von Operatoren, denn dadurch können zum Beispiel komplexe Zahlen mit dem "+" addiert werden.

```
complex& operator + (complex& x,
  complex& y) {return(complex(x.real +
  y.real, x.imag +y.imag)); }
```

Die Funktion *complex* ist ein Konstruktor und wird weiter unten erklärt. Wenn Sie nach dem obigen Beispiel alle Operatoren definieren, können Sie mit komplexen Zahlen rechnen, als wären diese vom Typ *float*:

```
complex a, b, c; ... a = b + c;
```

Da in C "+" für Strukturen nicht definiert ist, müßten Sie eine Funktion aufrufen, die die beiden Zahlen addiert:

```
a = add(b, c);
```

Eine deutliche Steigerung der Lesbarkeit also. Überladbar sind nahezu alle Operatoren, nicht jedoch deren Priorität und Assoziativität.

Unter anderem sind auch *new* und *delete* seit der Version 2.0 überladbar. Sie ermöglichen das dynamische Allozieren von Variablen; auch Felder und Objekte können alloziert werden. Sie stellen also eine Art Ersatz für Funktionen wie *malloc* und *free* dar, mit dem entscheidenden Vorteil, daß nicht der Speicher für zum Beispiel eine Variable angefordert wird, sondern die Variable selbst:

```
var_pointer = malloc(sizeof(int)); new int var;
  new charname[128]; delete name;
```

Aber widmen wir uns endlich den objektorientierten Neuerungen. Betrachten Sie die folgende Struktur:

```
struct RECHTECK { // Variablen int x, y, w,
  h; // Funktionen void retten(void); friend
  void bewegen(void); int löschen(void)
  { ... } }
```

Wie Sie sehen, wurde *struct* um die Fähigkeit erweitert, Funktionen innerhalb des Blockrumpfes zu deklarieren, sogar zu definieren. Der Vorteil ist, daß die Funktionen freien Zugriff auf die Variablen haben, die sich in derselben Struktur befinden, und zwar ohne lästige Operatoren wie "." oder "->" , sondern in derselben Art, wie dies bei lokalen Variablen geschieht. Wenn nun eine Instanz dieses Typs definiert wird, wird Platz für die Variablen und die Zeiger auf die Funktionen geschaffen. Die Funktionen selbst befinden sich natürlich nur einmal im Speicher und werden nicht mitkopiert. Eine Routine kann, wie oben schon erwähnt, wie eine Variable aufgerufen werden, zum Beispiel:

```
struct RECHTECK rechteck1;
      rechteck1.retten();
```

Die Deklaration einer Struktur ist oft der falsche Platz für Funktionsdefinitionen. Deswegen wurde ein Mechanismus entwickelt, der wie folgt aussieht:

```
void RECHTECK::retten(void) { ... }
```

Vor den Funktionsnamen kommt der Klassenname, getrennt durch " :: ". Auch hier kann man die Funktion wie oben geschildert aufrufen. Möchte man aber auf diese auch außerhalb zugreifen können, muß man sie als *friend* deklarieren und definieren. Damit *sagt* man dem Compiler, daß dies eine lose, nicht zur Struktur gehörende Funktion ist, die aber freien Zugriff auf die Variablen hat, die dort definiert wurden. Man hat nun freie Wahl, was den Aufruf dieser Funktion angeht:

```
rechteck1.bewegen(); bewegen();
```

Alle diese Mechanismen erweitern eine Struktur, bieten aber noch immer nicht die gewünschten Eigenschaften. Das Schlüsselwort *class* erweitert die Strukturen um Vererbung und alles sonstige, was noch fehlt! Betrachten Sie dazu Listing 1. Wie Sie sehen können, ähneln sich *struct* und *class*. Die Unterschiede machen das Salz in der Suppe aus. Da ist zunächst ein Doppelpunkt nach dem Klassennamen, der die optionalen Vaterklassen einleitet. Eine Vater- oder Superklasse gibt der Programmierer an, um dem Compiler anzudeuten, daß deren Eigenschaften an die hier beschriebene Klasse zu vererben sind. Man ist nicht auf einen Namen beschränkt, sondern kann Eigenschaften mehrerer Vaterklassen erben (multiple inheritance). Dabei kann man zusätzlich angeben, ob die Superklasse *privat* oder *public* vererbt wird. Damit hat es folgendes auf sich: die Verkapselung verbietet dem Programmierer in objektorientierten Programmiersprachen den Zugriff auf geschützte Bereiche. Geschützt sind zum Beispiel in Objective-C automatisch alle Variablen; die Methoden sind frei zugänglich. In C++ ist man jedoch weitergegangen und hat mit drei Schlüsselwörtern *public*, *private* und *protected* die Möglichkeit, die Zugriffsrechte frei zu bestimmen. Diese Schlüsselwörter bewirken, daß alle nachfolgenden Deklarationen und Definitionen in eine dieser Gruppen eingestuft werden. *public* steht für Bereiche, auf die sowohl aus einer Klasse heraus als auch von außerhalb zugegriffen werden kann. *private* und *protected* sind geschützte Bereiche, mit denen nur zugehörige Funktionen umgehen können. Vererbt man eine Klasse mit *public*, bleiben die Bereiche *public* und *protected* unverändert und werden so auch an den Nachfolger weitergegeben. *private*-Bereiche sind für den Nachkommen nicht sichtbar. Vererbt man die Klasse jedoch *private*, werden *public* und *protected privat*. Ein etwas komplexerer Zusammenhang, den man vereinfacht so darstellen könnte: *public* gehört allen, *protected* gehört mir und meinen Nachkommen, und *private* gehört nur mir.

Eine höchst interessante und sinnvolle Möglichkeit, die sich bei den Klassen anbietet, sind Konstruktoren und Destruktoren. Der Name impliziert es schon: ein Konstruktor erschafft. Er konstruiert eine Instanz einer Klasse. Sie können dazu eine oder mehrere (durch Überladung) Funktionen bereitstellen, die beim Entstehen

einer Variable dieser Klasse automatisch aufgerufen werden. Diese Funktionen haben denselben Namen wie die Klasse selbst. Im Listing 1 sind drei mögliche Konstruktoren definiert, die folgendermaßen benutzt werden können:

```
class Rechteck rechteck1; class Rechteck
   rechteck2(0, 0, desk_w, desk_h);
   class Rechteck rechteck3(rechteck2);
```

In der ersten Zeile wird eine Variable in üblicher Form definiert, zusätzlich werden aber auch die Klassenvariablen (in diesem Fall die Koordinaten des Rechtecks) auf 0 gesetzt. Die zweite Definition initialisiert mit ganz bestimmten Werten, und die dritte kopiert *rechteck2* nach *rechteck3*. Selbstverständlich können Konstruktoren auch bei dynamischer Definition angewandt werden:

```
new class Rechteck rechteck4(rechteck2);
```

In derselben Weise, in der Variablen zum Beispiel nach dem Beenden einer Funktion automatisch vom Stack gelöscht werden, geschieht dies auch bei den Klassen, wobei hier aber vor dem Löschen der Destruktor aufgerufen wird. Der Destruktor ist, ähnlich dem Konstruktor, eine Funktion mit dem Namen der Klasse, vor dem aber noch der Negierungsoperator, die Tilde (~) steht. Haben Sie in Ihren Konstruktoren zum Beispiel dynamisch Speicher alloziert, kann dieser hier automatisch gelöscht werden. Ein Destruktor hat selbstverständlich keine Argumente. Sehen Sie dazu Listing 2. Dort sehen Sie auch eine Variable *this*, die nirgends deklariert oder definiert wird. *this* ist ein Schlüsselwort und ein Zeiger auf das Objekt, dessen Funktion gerade abgearbeitet wird. Weiter unten, wenn Objective-C zur Diskussion stehen wird, werden wir uns eingehender damit beschäftigen. Dort heißt diese Variable *self*.

C++, so hat man sich entschieden, verwendet normalerweise das frühe Binden, das bedeutet, daß alle Aufrufe einer Funktion bereits beim Übersetzen bekannt sein müssen. Dies muß aber nicht immer der Fall sein: Nehmen wir zum Beispiel eine Klasse, der man beliebige Objekte unterschiedlicher Klassen übergeben kann. Die Objekte werden in der Klasse verwaltet und warten zum Beispiel auf einen Aufruf: *eachElementPerform:redraw*. Dieser Aufruf veranlaßt unsere Klasse, bei jedem Objekt *redraw* aufzurufen. Die Anwendungsmöglichkeiten einer solchen Klasse sind immens vielfältig und sollten auch in C++ verfügbar sein. Sie sind es auch! Virtuelle Funktionen ermöglichen das späte Binden und sind im Listing 2 daran zu erkennen, daß ihnen *virtual* vorangestellt ist. In der Praxis geht man folgendermaßen vor: Man definiert eine Klasse, die so gar nicht angewandt wird. Diese Klasse nennen wir abstrakt. Wir deklarieren bereits in dieser abstrakten Klasse Funktionen, die in ihren Nachfolgern definiert werden müssen. Man vererbt dadurch also einem ganzen *Stamm* die entsprechende Funktion, ohne sie eigentlich definiert zu haben. Sollte in einem Nachfolger die Funktion nicht implementiert sein, wird bei dessen Vater nachgeschaut. Sollte sie auch dort nicht definiert sein, wird das ganze wiederholt, bis man zur abstrakten Klasse selbst gekommen ist. Da sie dort auch nicht definiert ist, entsteht ein Laufzeitfehler. Um solchen Fehlern vorbeugen zu können, gibt es in C++ die Möglichkeit der virtuellen Funktionen:

```
virtual void beispiel(void) = 0;
```

Der Compiler wird sich weigern, eine Klasse, die solche rein virtuellen Funktionen enthält, zu instanzieren. Auch mit abgeleiteten Klassen wird es unmöglich sein, eine Variable zu definieren, solange sie keine Definitionen dieser Funktionen enthalten.

So, das waren die wichtigsten Eigenschaften von C++. Ich habe einige Details der Sprache einfach weggelassen, da sie den Überblick erschweren würden. Wie bereits gesagt, ist C++ noch nicht standardisiert, und es gibt noch keinen kommerziellen Compiler für den ST. Sollte sich an dieser Lage etwas ändern, werden wir das sofort berichten; auch würde dann einem Tutorial nichts mehr im Wege stehen. Die Eingefleischten und Speicherreichen können g++ von GNU benutzen, um sich erste Eindrücke von C++ zu verschaffen.

Objective-C

Ganz im Gegensatz zu C++ versucht Objective-C nicht, die bewährte Sprache in ihrer Syntax zu erweitern und zu verbessern, sondern gibt dem Programmierer einige Sprachkonstrukte, mit denen es möglich ist, objektorientiert zu programmieren. C ist hier nur das Mittel zum Zweck, denn auch Pascal, Fortran oder eine beliebige andere Programmiersprache hätte dazu benutzt werden können, diese Erweiterungen, die übrigens bewußt sehr an Smalltalk erinnern, zu tragen, wäre sie so flexibel wie C. Dadurch, daß sich diese Neuerungen auch optisch von dem restlichen Code unterscheiden, ist man in der Lage, Objektorientiertes sofort vom Konventionellen zu unterscheiden. Das ist wichtig, denn Objective-C führt die Erkenntnisse von Smalltalk knallhart ein: so wird von der Funktion Abstand genommen; man verschickt nunmehr Nachrichten an Objekte, auch wird nur das späte Binden angewandt. Anders als in C++ kann man hier die Objektorientierung nicht verringern, man hat aber noch immer die Chance, den Schwerpunkt zu verschieben. In seinem Buch, über das Sie in der Rubrik Buchbesprechungen mehr lesen können, beschreibt Brad Cox, wie Objective-C implementiert werden kann. Es ist aber bereits ein kommerzieller Precompiler für diese Sprache verfügbar, so daß dem Motto: „Programmieren geht über Studieren“ nicht mehr viel im Wege steht. Doch auch Objective-C ist noch in nicht in seiner Entwicklung abgeschlossen, so daß wir uns hier auf die Features beschränken, die in dem Buch erklärt und von dem uns zur Verfügung stehenden Compiler unterstützt wurden.

Objective-C erledigt die Verkapselung eleganterweise, indem es pro Datei eben nur eine einzige Objektdefinition erlaubt. Abbildung 3 zeigt eine solche Definition. Nach dem Anfangszeichen "=", das vor allem die Übersetzung erleichtert, folgt der Klassenname und der Name der Klasse, deren Eigenschaften geerbt werden sollen. Im Unterschied zu C++ ist die Vererbung nicht optional, sondern bindend. Alle Wege führen zu *Object*, dem Vater aller Klassen, der bereits wichtige Methoden enthält. Dem Vorgänger folgen Variablendefinitionen, wie bei einer C-Struktur, und denen wiederum die Liste aller Methoden, die mit "=:" abgeschlossen wird. In Objective-C gibt es zwei Arten von Methoden: die einen, zu erkennen am "+" zu Anfang der Definition, erzeugen Instanzen einer Klasse nach dem Prinzip der Konstruktoren in C++ und heißen Factory-Methoden; die anderen, die mit einem "-" beginnen, heißen Instance-Methoden und sind Methoden, die auf die Instanzen angewandt werden können. Der Unterschied erklärt sich am einfachsten mit einem Programmstück:

```
id myObject; myObject = [Object new];
                          [myObject free];
```

Dabei ist *new* eine Factory- und *free* eine Instance-Methode. Der Typ *id*, mit dem Objekte beliebiger Klasse belegt werden können, wird in derselben Art angewandt, wie das mit allen anderen C-Typen üblich ist. Das Versenden einer Nachricht geschieht wie in Smalltalk in zwei eckigen Klammern und verhält sich wie ein C-Ausdruck, d.h. daß er einen Wert zurückgibt, beliebig tief geschachtelt und aus anderen Ausdrücken aufgerufen werden kann. Der Typ des zurückgegebenen Wertes ist, wie die Parameter einer Nachricht, normalerweise *id*, kann aber mit einem C-Castoperator explizit umgewandelt werden. Aus einer Methode heraus

kann man auf alle Variablen, die in der Klasse definiert oder in die Klasse vererbt wurden, zugreifen, als seien sie lokal zu der Methode. Von außen jedoch hat man nur Zugriff auf die Methoden, die Variablen bleiben versteckt. Methoden, die vererbt wurden, können problemlos überladen oder ergänzt werden.

In dem obigen Beispiel, in dem *myObject* initialisiert wurde, gab die Methode *new* einen Wert des Typs *id* zurück. Damit *new* die Instanz, die sie gerade erzeugt hat, zurückgeben kann, braucht sie die Information über sich selbst. *self* ist das Objekt sebst. Das geht sogar so weit, daß Methoden, die in einem Objekt definiert sind, auf andere zugreifen können, die zum selben Objekt gehören:

```
- zeige { [self draw]; }
```

Sollte eine Methode nicht in der Klasse selbst definiert sein, wird, wie in C++, beim Vater nachgeschaut. Sollte die Methode nicht zu finden sein, wird eine Fehlerroutine abgearbeitet.

Es passiert häufig, daß vererbte Methoden nicht mehr ganz ihrem Zweck nachkommen. Es wäre sinnlos, diese durch vollständiges Überladen ganz wegzuwerfen. Man möchte also eine Methode der Vaterklasse verwenden, obwohl man sie überladen hat. Die Factory-Methode *new* zum Beispiel wird oft zu Demonstrationszwecken herangezogen:

```
+ new { return([super new]); }
```

super bezieht sich auf den Vater und vermeidet Rekursion, die auftreten würde, wenn man *self* verwenden würde. Die neue Methode kann noch zusätzlich ergänzt werden. In unserem Beispiel könnte man die Variablen initialisieren.

Die Methoden, die hier aufgeführt wurden, verwenden keine Argumente. Aber auch diese können verwendet werden. Die Methode *point* aus der Abbildung 3 benutzt Argumente. Damit Sie ein Gefühl für das Aussehen der Sprache bekommen, habe ich ein kurzes Listing zusammengestellt (Listing 3).

Wie bereits weiter oben erwähnt, ist die Basisklasse von Objective-C das *Object*.

```
= Klassenname: Vaterklasse
{
        // Variablen im C-Stil
}

+Factorymethode
{
        // Definition
}

-Instancemethode
{
        // Aufruf einer Factorymethode
        [Klassenname Factorymethode];

        // Aufruf einer Instancemethode
        [Objektname Instancemethode];

        // Aufruf einer Methode mit Variablen
        [Objektname pointX: 0 Y: 0 Color: 2];
}

-pointX: (int) sx Y: (int) sy Color: (int) sc
{
        // Definition
}

=:
```

Abb.3: Die Methode Point

Object ist eine Klasse aus dem Standardpaket. Die Idee hinter OOP ist ja, daß vorhandene und ausgetestete Klassen an den Kunden geliefert werden, der diese erweitern und ergänzen kann, ähnlich wie Elektroniker das mit ihren ICs machen. Dort wird Objektorientierung scheinbar schon lange sehr erfolgreich angewandt. Das soll nun auch mit der Software möglich sein (Software-IC); nur wenn keine Klassen existieren, kann man diese auch nicht erweitern. Deshalb liefert die Firma Stepstone, die die Rechte an Objective-C besitzt und auch einen Compiler produziert, ein Paket sinnvoller Klassen mit ihrem Produkt. Diese enthalten unter anderem komfortable Felder, Collections ähnlich der oben beschriebenen, und Mengen.

Zusammenfassend könnte man sagen, daß Objective-C die Objektorientierung zielstrebiger verfolgt als C++. Letztere Sprache hat aber entscheidende Vorteile; beispielsweise ist der Name nicht rechtlich geschützt. Dadurch existieren auf den Kompatiblen, Unix und dem Mac bereits viele unterschiedliche Compiler. Viele Programmierer betrachten C++ als würdigen Nachfolger von C und steigen deshalb um. Da aber C++ eine umfangreiche Sprache ist und in vielen Punkten von der C-Ideologie abweicht, stellt sich für mich die Frage, ob es denn nicht sinnvoller wäre, gleich auf eine *richtige* objektorientierte Sprache wie Smalltalk oder Eiffel umzusteigen, wenn man denn nun überhaupt objektorientiert programmieren will.

Grischa Ekart

Literatur:

OOP allgemein:
A. Winblad, S. Edward, Object-Oriented Programming, Addison Wesley 1990, ISBN 0-201-50736-6

C++:
B. Stroustrup, Die C++ Programmiersprache, Addison Wesley 1987, ISBN 3-92511872-1

S. Lippman, C++ Einführung und Leitfaden, Addison Wesley 1990, ISBN 3-89319-276-6

Objective-C:
Brad J. Cox, Object-Oriented Programming, Addison Wesley 1986, ISBN 0-201-10393-1

```
 1:  class   Rechteck
 2:  {
 3:  protected:
 4:       int      x;
 5:       int      y;
 6:       int      w;
 7:       int      h;
 8:  public:
 9:       rechteck(void);
10:       rechteck(int rx, int ry, int rw, int rh);
11:       rechteck(Rechteck& r);
12:       ~rechteck(void);
13:
14:          void    retten(void);
15:          friend  void bewegen(void);
16:          int     löschen(void)
17:          {
18:               // Funktionsdefinition...
19:          }
20:  };
21:
22:  Rechteck::rechteck(void)
23:  {
24:       x = 0;
25:       y = 0;
26:       w = 0;
```

→

```
27:     h = 0;
28: }
29:
30: Rechteck::rechteck(int rx, int ry, int rw, int rh)
31: {
32:     x = rx;
33:     y = ry;
34:     w = rw;
35:     h = rh;
36: }
37:
38: Rechteck::rechteck(Rechteck& r)
39: {
40:     x = r.x;
41:     y = r.y;
42:     w = r.w;
43:     h = r.h;
44: }
45:
46: Rechteck::~rechteck(void)
47: {
48: }
49:
50: void
51: Rechteck::retten(void)
52: {
53:     // Funktionsdefinition...
54: }
55:
56: friend void
57: bewegen(void)
58: {
59:     // Funktionsdefinition...
60: }
```

```
 1: class   Fenster: public Rechteck
 2: {
 3: protected:
 4:     int     art;
 5:     char    *fenster_name;
 6: public:
 7:     fenster(char *name)
 8:     ~fenster(void);
 9:     virtual redraw(void) = 0;
10:     virtual Fenster& öffne(void);
11: }
12:
13: Fenster::fenster(char *name)
14: {
15:     fenster_name = malloc(strlen(name) + 1);
16:     strcpy(fenster_name, name);
17: }
18:
19: Fenster::~fenster(void)
20: {
21:     free(fenster_name);
22: }
23:
24: Fenster& Fenster::öffne(void)
25: {
26:     // Funktionsdefinition
27:     return(*this);
28: }
```

```
 1: extern  id Object;
 2:
 3: = Kunde : Object
 4: {
 5:     char    firma[40];
 6:     char    vorname[40];
 7:     char    nachname[40];
 8:     char    straPe[40];
 9:     int     plz;
10:     char    ort[40];
11: }
12:
13: + anlegen
14: {
15:     char    buffer[40];
16:
17:     self = [super new];
18:     printf("Neuer Kunde:\n\n");
19:     printf("Firma: ");
20:     gets(firma);
21:     printf("Vorname: ");
22:     gets(vorname);
23:     printf("Name: ");
24:     gets(name);
25:     printf("StraPe: ");
26:     gets(straPe);
27:     printf("Postleitzahl: ");
28:     gets(buffer);
29:     plz = atoi(buffer);
30:     printf("Ort: ");
31:     gets(ort);
32:     return(self);
33: }
34:
35: - (void)löschen
36: {
37:     [self free];
38: }
39:
40: - (void)ausgeben
41: {
42:     printf("%s\n%s\n%s\n%s\n%d\n%s\n",
43:         firma,
44:         vorname,
45:         nachname,
46:         straPe,
47:         plz,
48:         ort);
49: }
50:
51: =:
52:
53: int main(void);
54:
55: int
56: main(void)
57: {
58:     id neu;
59:
60:     id = [Kunde anlegen];
61:     [neu ausgeben];
62:     [neu löschen];
63:     return(0);
64: }
```

Programmer's Toolbox - Dateien

Teil 13: Einfache Verschlüsselungsverfahren



© M.V. ZIMMERMANN

In der heutigen Folge beginnt der dritte und letzte thematische Block der Programmer's Toolbox. Hierbei geht es ausschließlich um das Verschlüsseln. Es werden Kommandos zum Verschlüsseln von ganzen Dateien implementiert. Darüber hinaus werden einige Kommandos eingeführt, mit deren Hilfe sich eine Benutzerverwaltung mit Paßwortschutz aufbauen läßt.

Vom Verschlüsseln

Der dritte thematische Block umfaßt insgesamt drei Serienteile. Im folgenden sehen Sie eine kurze Übersicht.

In der heutigen Folge (13. Teil) stelle ich Ihnen drei Kommandos zum Ver- und Entschlüsseln von ganzen Dateien vor. Hierbei handelt es sich um:

CAESAR -	ein einfaches Verschlüsselungsverfahren
POLY -	polyalphabetische Verschlüsselung
GEO -	Verschlüsselung durch Permutation mit Hilfe einer Matrix

In der nächsten Folge (14. Teil) kommen wir zum Data Encryption Standard (DES), und es wird gezeigt, wie man mit dem DES eine Benutzerverwaltung mit Paßwortschutz aufbauen kann. Die letzte Folge dieser Serie (15. Teil) führt dann vier Kommandos ein, die auf der Benutzerverwaltung des vorhergehenden Teils aufbauen:

MKUSER -	Anlegen eines Benutzerdatensatzes
RMUSER -	Löschen eines Benutzerdatensatzes
PASSWD -	Ändern von Paßwörtern
LOGIN -	Einloggen in das System

Einige Verschlüsselungsverfahren

Soviel zur Einstimmung. Es folgen konkrete Verschlüsselungsverfahren, implementiert innerhalb der drei Kommandos CAESAR, POLY und GEO. Die Dokumentation erfolgt im Rahmen des Schemas, welches bereits bei den früheren Kommandos verwendet worden ist.

Name

CAESAR - ein einfaches Verschlüsselungsverfahren

Anwendung

CAESAR [-C I -E] Buchstabe

Beschreibung

Das Kommando CAESAR realisiert einen einfachen Algorithmus zum Ver- bzw. Entschlüsseln von Dateien, der angeblich bereits von Cäsar verwendet worden ist. Auf den Text einer Botschaft wird ein bestimmter Buchstabe „aufaddiert", wodurch man die Botschaft unkenntlich macht.

Beim Kommando CAESAR wird der Schlüsselbuchstabe als Parameter übergeben. Die Botschaft wird aus dem Standardeingabekanal gelesen und in den Standardausgabekanal geschrieben.

Optionen

-C Verschlüsseln einer Botschaft. Die Option -C ist voreingestellt. Sie muß also nicht explizit notiert werden.

-E Entschlüsseln der Botschaft

Beispiel

Im folgenden Beispiel wird eine winzige Datei mit dem CAESAR-Verfahren zunächst ver- und dann wieder entschlüsselt, wobei die Zwischenergebnisse jeweils ausgegeben werden.

```
$ #
$ # Erzeugen und Anzeigen der Datei
    KLARTEXT
$ #
$ ECHO "Dies ist eine geheime
        Botschaft." > KLARTEXT
$ CAT KLARTEXT

Dies ist eine geheime Botschaft.

$ #
$ # Verschlüsseln und Anzeigen von
    CRYPTEXT
$ #
$ CAESAR -C F < KLARTEXT > CRYPTEXT
$ CAT CRYPTEXT

è»½"f»"'f½»œ½f¡½«½»|½fêŒ'"–«º
¼'tfP

$ #
$ # Entschlüsseln von CRYPTEXT
$ #
$ CAESAR -E F < CRYPTEXT
```

Implementierung

Die Implementierung des Kommandos CAESAR ist recht einfach (Listing 3.1). Im wesentlichen umfaßt sie die beiden Funktionen *crypt* und *encrypt*, die das Ver- bzw. Entschlüsseln realisieren. In *crypt* wird auf die Daten aus der Standardeingabe der Schlüsselbuchstabe aufaddiert. In *encrypt* wird dagegen umgekehrt verfahren.

Die Übersetzungsinformationen werden wie bisher mit Hilfe eines Make-Files notiert (Listing MAKE). Das entsprechende Listing enthält die Informationen für

sämtliche Module und Kommandos des dritten Blocks der Programmer's Toolbox.

Name
POLY - polyalalphabetische Verschlüsselung

Anwendung
POLY [-C | -E] Wort

Beschreibung
Das im Kommando CAESAR verwendete Verfahren ist offensichtlich nicht besonders sicher. Zu Zeiten Cäsars mag es wohl ausgereicht haben, aber geht man davon aus, daß nur 256 Schlüsselbuchstaben existieren (der Zeichensatz des ST umfaßt 256 Zeichen), ist das Knacken dieses Codes mit Leichtigkeit zu bewerkstelligen. Einen höheren Sicherheitsgrad erlangt man durch die Verwendung mehrerer Buchstaben, die abwechselnd auf den Klartext aufaddiert werden. Es ergibt sich die sogenannte polyalphabetische Verschlüsselung. Sie wird mit dem Kommando POLY realisiert. Als Schlüssel wird POLY ein Wort übergeben, dessen Komponenten (Buchstaben) abwechselnd auf die Buchstaben des Klartext „aufaddiert" werden.

Optionen
Die Optionen des Kommandos POLY entsprechen den Optionen von CAESAR.

Beispiel
Auch für die Anwendung des Kommandos POLY soll mit dem folgenden ein Beispiel gegeben werden:

```
$ #
$ # Verschlüsseln und Anzeigen von
    CRYPTEXT
$ #
$ POLY -C ATARI < KLARTEXT > KRYPTEXT
$ CAT KRYPTEXT

à©ª–i¬|Œr«–xªrãª¶ª†Áªtâij©œÁ–|»Œéa\

$ #
$ # Entschlüsseln von CRYPTEXT
$ #
$ POLY -E ATARI < KRYPTEXT

Dies ist eine geheime Botschaft.
```

Implementierung
Die Implementierung von POLY (Listing 3.2) entspricht weitgehend der von CAESAR (Listing 3.1). Innerhalb der Funktionen crypt und encrypt befinden sich nun lediglich zusätzliche Anweisungen, die eine Indizierung der Buchstaben des Schlüsselworts bewerkstelligen. Dadurch sind es immer andere Schlüsselbuchstaben, die auf die Botschaft „aufaddiert" werden.

Name
GEO - Verschlüsselung durch Permutation mit Hilfe einer Matrix

Anwendung
GEO [-C | -E] Zahl Zahl

Beschreibung
Mit POLY besitzt man nun bereits eine ganze Reihe von Variationsmöglichkeiten. Wortlänge und Wortinhalt des Schlüssels können variieren. Die Anzahl der möglichen Schlüssel steigt deutlich. POLY und CAESAR gehören dabei zu einer Klasse von Verschlüsselungsverfahren, die eine Verschlüsselung durch Verfremdung der Buchstaben einer Botschaft erreichen. Eine andere Möglichkeit besteht in der Permutation der Buchstaben einer Botschaft, d.h. die Positionen der Buchstaben werden nach bestimmten Regeln geändert. Von dieser Möglichkeit ist im folgenden Gebrauch gemacht.

Innerhalb des Kommandos GEO wird eine Verschlüsselung mit Hilfe einer Matrix vorgenommen. Die Botschaft wird in einer bestimmten Reihenfolge in die Matrix eingebracht und in einer anderen Reihenfolge wieder aus ihr ausgelesen. Dadurch ergibt sich eine Permutation der Botschaft.

Dieses Verschlüsselungsverfahren kann mit Hilfe der Matrixabmessungen parametrisiert werden. Entsprechend erwartet das Kommando GEO zwei positive ganze Zahlen als Schlüsselparameter.

Optionen
Die Optionen des Kommandos GEO entsprechen denen von CAESAR und POLY.

Beispiel
Das nachfolgende Beispiel für das Kommando GEO belegt, daß im Gegensatz zu den vorherigen Verschlüsselungen bei GEO eine Permutation der Botschaft stattfindet. - Die Buchstabenhäufigkeit der verschlüsselten Botschaft entspricht der der unverschlüsselten Botschaft.

```
$ #
$ # Verschlüsseln und Anzeigen von
    CRYPTEXT
$ #
$ GEO -C 6 7 < KLARTEXT > KRYPTEXT
$ CAT KRYPTEXT

.c hnD hBesi
aoetefi  stmge tseeii

$ #
$ # Entschlüsseln von CRYPTEXT
$ #
$ GEO -E 6 7 < KRYPTEXT

Dies ist eine geheime Botschaft.
```

Vorausschau

Die drei in der heutigen Folge betrachteten Verschlüsselungsverfahren besitzen einen entscheidenden Nachteil: Sie sind sehr einfach. Die ersten beiden Verfahren verschlüsseln lediglich die Buchstaben der Botschaft. Die Positionen bleiben dabei erhalten. Das dritte Verfahren behält die Buchstaben bei und modifiziert lediglich die Positionen derselben. Alle drei Verfahren sind entsprechend relativ leicht zu „knacken".

In der nächsten Folge wird ein Verschlüsselungsverfahren abgelistet, das sich nicht so leicht knacken läßt. Dabei handelt

```
 1:  ####################################################
 2:  # Listing MAKE Datei : MAKE3                       #
 3:  # Modifikationsdatum : 8-Jan-91                    #
 4:  # Abhängigkeiten     : -                           #
 5:  ####################################################
 6:
 7:  COMPILER  = \megamax\ccom.ttp -I\megamax\headers
 8:  LINKER    = \megamax\ld.ttp \megamax\init.o
 9:  PROGRAMM3 = caesar.ttp poly.ttp geo.ttp
             mkuser.ttp \
10:              rmuser.ttp passwd.ttp login.ttp
11:  MODUL3    = crypt.o usermain.o
12:
13:  make_3 : $(PROGRAMM3) $(MODUL3)
14:
15:  ####################################################
16:  # Teil 3 - Vom Verschlüsseln                       #
17:  ####################################################
18:
19:  caesar.ttp : caesar.c
20:  $(COMPILER) caesar.c
21:  $(LINKER) caesar.o -lc -o caesar.ttp
22:
23:  poly.ttp : poly.c
24:  $(COMPILER) poly.c
25:  $(LINKER) poly.o -lc -o poly.ttp
26:
27:  geo.ttp : geo.c
28:  $(COMPILER) geo.c
29:  $(LINKER) geo.o -lc -o geo.ttp
30:
31:  mkuser.ttp : mkuser.c usermain.h usermain.o    →
```

```
32:  $(COMPILER) mkuser.c
33:  $(LINKER) mkuser.o usermain.o -lc -o mkuser.ttp
34:
35:  rmuser.ttp : rmuser.c usermain.h usermain.o
36:  $(COMPILER) rmuser.c
37:  $(LINKER) rmuser.o usermain.o -lc -o rmuser.ttp
38:
39:  passwd.ttp : passwd.c usermain.h usermain.o
     crypt.h crypt.o
40:  $(COMPILER) passwd.c
41:  $(LINKER) passwd.o usermain.o crypt.o -lc -o
     passwd.ttp
42:
43:  login.ttp : login.c usermain.h usermain.o
     crypt.h crypt.o
44:  $(COMPILER) login.c
45:  $(LINKER) login.o usermain.o crypt.o -lc -o
      login.ttp
46:
47:  crypt.o : crypt.c
48:  $(COMPILER) crypt.c
49:
50:  usermain.o : usermain.c
51:  $(COMPILER) usermain.c
```

```
  1:  /*
  2:   * Listing 3.1, Datei : caesar.c
  3:   * Programm           : CAESAR - Ein einfaches
  4:   *                      Verschlüsselungsverfahren
  5:   * Modifikationsdatum : 14-Juli-1990
  6:   * Abhängigkeiten     : stdio.h, local.h
  7:   */
  8:
  9:  #include <stdio.h>
 10:  #include "local.h"
 11:
 12:  /*
 13:   * Funktionen    : crypt, encrypt
 14:   *
 15:   * Parameter     : crypt(key);
 16:   *                 encrypt(key);
 17:   *                 unsigned char key;
 18:   *
 19:   * Aufgabe       :
 20:   *
 21:   * Ver- und Entschlüsselung mit dem CAESAR-
       Verfahren.
 22:   * Der Schlüssel <key> wird als Parameter
       übergeben.
 23:   * Als Ein- und Ausgabe dienen die beiden
 24:   * Standardkanäle.
 25:   */
 26:
 27:  void crypt(key)
 28:  unsigned char key;
 29:  {   short help;
 30:
 31:      while (!feof(stdin)) {
 32:          help = getchar() + key;
 33:          if (help > 255)
 34:              help -= 256;
 35:          if (!feof(stdin))
 36:              putchar((unsigned char)help);
 37:      }
 38:  }
 39:
 40:  void encrypt(key)
 41:  unsigned char key;
 42:  {   short help;
 43:
 44:      while (!feof(stdin)) {
 45:          help = getchar() - key;
 46:          if (help < 0)
 47:              help += 256;
 48:          if (!feof(stdin))
 49:              putchar((unsigned char)help);
 50:      }
 51:  }
 52:
 53:  /*
 54:   * Funktion      : caesar
 55:   *
 56:   * Parameter     : ok = caesar(argc, argv);
 57:   *                 BOOLEAN ok;
 58:   *                 short   argc;
 59:   *                 char    *argv[];
 60:   *
 61:   * Aufgabe       :
 62:   *
 63:   * Interpretation der durch <argc> und <argv>
 64:   * spezifizierten Parameterliste gemäß den Fest-
 65:   * legungen des Kommandos CAESAR.
 66:   */
 67:
 68:  BOOLEAN caesar(argc, argv)
 69:  short argc;
 70:  char *argv[];
 71:  {   unsigned char key;
 72:
 73:      if (argc == 3) {
 74:          if (strlen(argv[2]) == 1)
 75:              key = argv[2][0];
 76:          else {
 77:              fprintf(stderr,
 78:          "caesar: command expects a letter for
              key\n");
 79:              return(FALSE);
 80:          }
 81:          if (strcmp(argv[1], "-e") == 0 ||
 82:              strcmp(argv[1], "-E") == 0) {
 83:              encrypt(key);
 84:              return(TRUE);
 85:          }
 86:          else if (strcmp(argv[1], "-c") == 0 ||
 87:                   strcmp(argv[1], "-C") == 0) {
 88:              crypt(key);
 89:              return(TRUE);
 90:          }
 91:          else {
 92:              fprintf(stderr,
 93:                "caesar: option -c or -e
                   expected\n");
 94:              return(FALSE);
 95:          }
 96:      }
 97:      else {
 98:          fprintf(stderr, "%s\n%s\n",
 99:                "SYNOPSIS: caesar -c key",
100:                "          caesar -e key");
101:          return(FALSE);
102:      }
103:  }
104:
105:  void main(argc, argv)
106:  short argc;
107:  char *argv[];
108:  {   if (!caesar(argc, argv))
109:          exit(1);
110:      exit(0);
111:  }
```

```
  1:  /*
  2:   * Listing 3.2, Datei : poly.c
  3:   * Programm           : POLY - Polyalphabetische
  4:   *                      Verschlüsselung
  5:   * Modifikationsdatum : 14-Juli-1990
  6:   * Abhängigkeiten     : stdio.h, local.h
  7:   */
  8:
  9:  #include <stdio.h>
 10:  #include "local.h"
 11:
 12:  /*
 13:   * Funktionen    : crypt, encrypt
 14:   *
 15:   * Parameter     : crypt(key);
 16:   *                 encrypt(key);
 17:   *                 unsigned char *key;
 18:   *
 19:   * Aufgabe       :
 20:   *
 21:   * Ver- und Entschlüsselung mit einem
 22:   * polyalphabetischen Verfahren. <key> ist ein
 23:   * Zeiger auf eine nullterminierte Zeichenkette.
 24:   */
 25:
 26:  void crypt(key)
 27:  unsigned char *key;
 28:  {   short help,
```

→

```
 29:         i = 0,
 30:         l = strlen(key);
 31:
 32:     while (!feof(stdin)) {
 33:         help = getchar() + key[i];
 34:         if (help > 255)
 35:             help -= 256;
 36:         if (!feof(stdin))
 37:             putchar((unsigned char)help);
 38:         i++;
 39:         if (i == l)
 40:             i = 0;
 41:     }
 42: }
 43:
 44: void encrypt(key)
 45: unsigned char *key;
 46: {   short help,
 47:         i = 0,
 48:         l = strlen(key);
 49:
 50:     while (!feof(stdin)) {
 51:         help = getchar() - key[i];
 52:         if (help < 0)
 53:             help += 256;
 54:         if (!feof(stdin))
 55:             putchar((unsigned char)help);
 56:         i++;
 57:         if (i == l)
 58:             i = 0;
 59:     }
 60: }
 61:
 62: /*
 63:  * Funktion     : poly
 64:  *
 65:  * Parameter    : ok = poly(argc, argv);
 66:  *                BOOLEAN ok;
 67:  *                short   argc;
 68:  *                char    *argv[];
 69:  *
 70:  * Aufgabe      :
 71:  *
 72:  * Interpretation der durch <argc> und <argv>
 73:  * spezifizierten Parameterliste gemäß den Fest-
 74:  * legungen des Kommandos POLY.
 75:  */
 76:
 77: BOOLEAN poly(argc, argv)
 78: short argc;
 79: char  *argv[];
 80: {   if (argc == 3) {
 81:         if (strcmp(argv[1], "-e") == 0 ||
 82:             strcmp(argv[1], "-E") == 0) {
 83:             encrypt(argv[2]);
 84:             return(TRUE);
 85:         }
 86:         else if (strcmp(argv[1], "-c") == 0 ||
 87:                  strcmp(argv[1], "-C") == 0) {
 88:             crypt(argv[2]);
 89:             return(TRUE);
 90:         }
 91:         else {
 92:             fprintf(stderr,
 93:                     "poly: option -c or -e
                         expected\n");
 94:             return(FALSE);
 95:         }
 96:     }
 97:     else {
 98:         fprintf(stderr, "%s\n%s\n",
 99:                 "SYNOPSIS: poly -c key",
100:                 "          poly -e key");
101:         return(FALSE);
102:     }
103: }
104:
105: void main(argc, argv)
106: short argc;
107: char  *argv[];
108: {   if (!poly(argc, argv))
109:         exit(1);
110:     exit(0);
111: }
112:
```

```
  1: /*
  2:  * Listing 3.3, Datei : geo.c
  3:  * Programm           : GEO - Verschlüsselung durch
  4:  *                      Permutation mit Hilfe einer
  5:  *                      Matrix
  6:  * Modifikationsdatum : 15-Juli-1990
  7:  * Abhängigkeiten     : stdio.h, local.h
  8:  */
  9:
 10: #include <stdio.h>
 11: #include "local.h"
 12:
 13: /*
 14:  * Funktionen   : crypt, encrypt
 15:  *
 16:  * Parameter    : crypt(x, y);
 17:  *                encrypt(x, y);
 18:  *                short x, y;
 19:  *
 20:  * Aufgabe      :
 21:  *
 22:  * Ver- und Entschlüsselung mit einem
 23:  * geometrischen Verfahren. Die Eingabedaten werden
 24:  * in eine Matrix eingetragen, deren Größe durch <x>
 25:  * und <y> parametrisiert ist.
 26:  */
 27:
 28: void crypt(x, y)
 29: short x, y;
 30: {   short max_char,
 31:           akt_char,
 32:           xi,
 33:           yi,
 34:           px,
 35:           py;
 36:     char  *infield,
 37:           help;
 38:
 39:     max_char = x * y;
 40:     infield = malloc(max_char);
 41:     do {
 42:         akt_char = 0;
 43:         do {
 44:             help = getchar();
 45:             if (!feof(stdin)) {
 46:                 infield[akt_char] = help;
 47:                 akt_char++;
 48:             }
 49:         } while(!feof(stdin) && akt_char <
              max_char);
 50:         for (xi = 0; xi < x; xi++)
 51:             for (yi = 0; yi < y; yi++) {
 52:                 px = (xi + yi) % x;
 53:                 py = y - yi - 1;
 54:                 if (py * x + px < akt_char)
 55:                     putchar(infield[py * x +
                          px]);
 56:             }
 57:     } while (!feof(stdin));
 58:     free(infield);
 59: }
 60:
 61: void encrypt(x, y)
 62: short x, y;
 63: {   short max_char,
 64:           akt_char,
 65:           xi,
 66:           yi,
 67:           px,
 68:           py,
 69:           i;
 70:     char  *infield,
 71:           *outfield,
 72:           help;
 73:
 74:     max_char = x * y;
 75:     infield = malloc(max_char);
 76:     outfield = malloc(max_char);
 77:     do {
 78:         akt_char = 0;
 79:         do {
 80:             help = getchar();
 81:             if (!feof(stdin)) {
 82:                 infield[akt_char] = help;
 83:                 akt_char++;
 84:             }
 85:         } while(!feof(stdin) && akt_char <
              max_char);
```

→

```
 86:         i = 0;
 87:         for (xi = 0; xi < x; xi++)
 88:             for (yi = 0; yi < y; yi++) {
 89:                 px = (xi + yi) % x;
 90:                 py = y - yi - 1;
 91:                 if (py * x + px < akt_char)
 92:                     outfield[py * x + px] =
 93:                     infield[i++];
 94:             }
 95:         for (i = 0; i < akt_char; i++)
 96:             putchar(outfield[i]);
 97:     } while (!feof(stdin));
 98:     free(infield);
 99:     free(outfield);
100: }
101:
102: /*
103:  * Funktion     : geo
104:  *
105:  * Parameter    : ok = geo(argc, argv);
106:  *                BOOLEAN ok;
107:  *                short   argc;
108:  *                char    *argv[];
109:  *
110:  * Aufgabe      :
111:  *
112:  * Interpretation der durch <argc> und <argv>
113:  * spezifizierten Parameterliste gemäß den Fest-
114:  * legungen des Kommandos GEO.
115:  */
116:
117: BOOLEAN geo(argc, argv)
118: short argc;
119: char  *argv[];
120: {   short x, y;
121:
122:     if (argc == 4) {
123:         x = atoi(argv[2]);
124:         y = atoi(argv[3]);
125:         if (x <= 0 || y <= 0) {
126:             fprintf(stderr,
127:        "geo: the keys have to be positive
            integers\n");
128:             return(FALSE);
129:         }
130:         if (strcmp(argv[1], "-e") == 0 ||
131:             strcmp(argv[1], "-E") == 0) {
132:             encrypt(x, y);
133:             return(TRUE);
134:         }
135:         else if (strcmp(argv[1], "-c") == 0 ||
136:                  strcmp(argv[1], "-C") == 0) {
137:             crypt(x, y);
138:             return(TRUE);
139:         }
140:         else {
141:             fprintf(stderr,
142:                 "geo: option -c or -e
                     expected\n");
143:             return(FALSE);
144:         }
145:     }
146:     else {
147:         fprintf(stderr, "%s\n%s\n",
148:                 "SYNOPSIS: geo -c key1 key2",
149:                 "          geo -e key1 key2");
150:         return(FALSE);
151:     }
152: }
153:
154: void main(argc, argv)
155: short argc;
156: char  *argv[];
157: {   if (!geo(argc, argv))
158:         exit(1);
159:     exit(0);
160: }
```

PRO LOGIK

Prolog für Einsteiger Teil 2

Nach dem Schnupperkurs der letzten Folge wollen wir dieses Mal richtig in die Programmierung der Sprache Prolog einsteigen. Das letzte Mal wurde schon erwähnt, daß Prolog eine logische Programmiersprache ist. Dieser logische Ursprung hat natürlich Auswirkungen auf die Funktionsweise von Prolog und die Art und Weise, in der man Prolog-Programme erstellt. Um diese Besonderheiten und ihre Grundlagen geht es in dieser zweiten Folge der Prolog-Serie.

Wir beschäftigen uns zunächst mit den Prolog-Mechanismen, die ihren Ursprung in dem formalen Hintergrund haben, aus dem Prolog entstanden ist. Danach wird die anwendungsorientierte Seite beleuchtet, deren Ursprung keine theoretischen Überlegungen, sondern praktische Bedürfnisse, sind. Dabei werden auch die wichtigsten eingebauten Prädikate vorgestellt, die in jedem Prolog-System vorhanden sind.

Back to the roots

Zunächst aber ein kleiner Exkurs zu den Wurzeln der Logikprogrammierung. Der logische Formalismus, der den Ursprung von Prolog bildet, ist eine Einschränkung der Prädikatenlogik erster Ordnung, die sogenannte Hornlogik. Die Hornlogik erlaubt es, Aussagen über ein Problem mit Hilfe von Klauseln zu machen. Diese Klauseln werden, wie in Prolog (siehe letzte Folge), in Fakten und Regeln unterteilt. Interessant ist nun, welche neuen Klauseln aus den bereits vorhandenen gefolgert werden können. Um die dahintersteckende Idee etwas zu verdeutlichen, schauen wir uns ein einfaches Beispiel an. Wir gehen davon aus, daß alle Äpfel eßbar sind und es ein bestimmtes Objekt gibt, das die Bezeichnung „roter Apfel" trägt und natürlich ein Apfel ist. Dieser Sachverhalt kann in Hornlogik, unter Verwendung der aus der letzten Folge bekannten Prolog-Notation, etwa so ausgedrückt werden:

```
essbar(X) :- apfel(X).
apfel(roterApfel).
```

Die erste Klausel (eine Regel) lautet: *„X ist eßbar, falls X ein Apfel ist"*. Das Zeichen ':-' wird dabei als „falls" gelesen. Die zweite Klausel (ein Fakt) kann man als *„'roterApfel' ist ein Apfel"* lesen. Es ist sehr wichtig, dabei zu beachten, daß die Variable (beginnt mit einem Großbuchstaben) „X" für jedes beliebige Objekt stehen kann, während das Atom *roterApfel* für ein einziges Objekt steht. Unter den hier geschilderten Bedingungen wird ein Mensch recht schnell zu dem Schluß kommen, daß dieses Objekt *roter Apfel* wohl eßbar ist, denn schließlich ist es ein Apfel, und alle Äpfel sind, nach der vorliegenden Regel, eßbar. Diese Einsicht fällt einem Computer bedeutend schwerer.

Interessant ist in diesem Zusammenhang natürlich ein Programm, das feststellt, ob eine gegebene Behauptung aus einer Menge von Klauseln gefolgert werden kann. In Prolog wird die Frage, ob das Objekt *roter Apfel* eßbar ist, wie folgt gestellt:

```
?- essbar(roterApfel).
```

Nach der Eingabe einer solchen Anfrage muß das Prolog-System feststellen, ob die Anfrage aus den vorhandenen Regeln gefolgert werden kann. Dies tut es mit einem Algorithmus, der auf dem 1965 von Robinson, im Rahmen der Entwicklung von

Systemen zum automatischen Beweisen, gefunden Resolutionprinzips aufbaut. Obwohl wir die Resolution hier nicht im Detail besprechen wollen, werden wir uns doch ein wenig mit der Funktionsweise von Prolog-Interpretern beschäftigen, soweit dies für das Verständnis von Prolog-Programmen hilfreich ist.

Einer der Gründe, warum die Resolution so einfach von Rechnern ausgeführt werden kann, ist, daß sie nur aus einer einzelnen Regel besteht. Diese heißt naheliegenderweise Resolutionsregel und lautet wie folgt: Kommt in einer Anfrage ein Prädikat *p(A1, ..., An)* vor, so kann es durch die rechte Seite einer Klausel ersetzt werden, falls deren linke Seite zu dem Prädikat paßt, wobei *A1* bis *An* für die Argumente des Prädikats steht. Unter welchen Umständen ein Prädikat zur linken Seite einer Klausel paßt, werden wir später noch sehen.

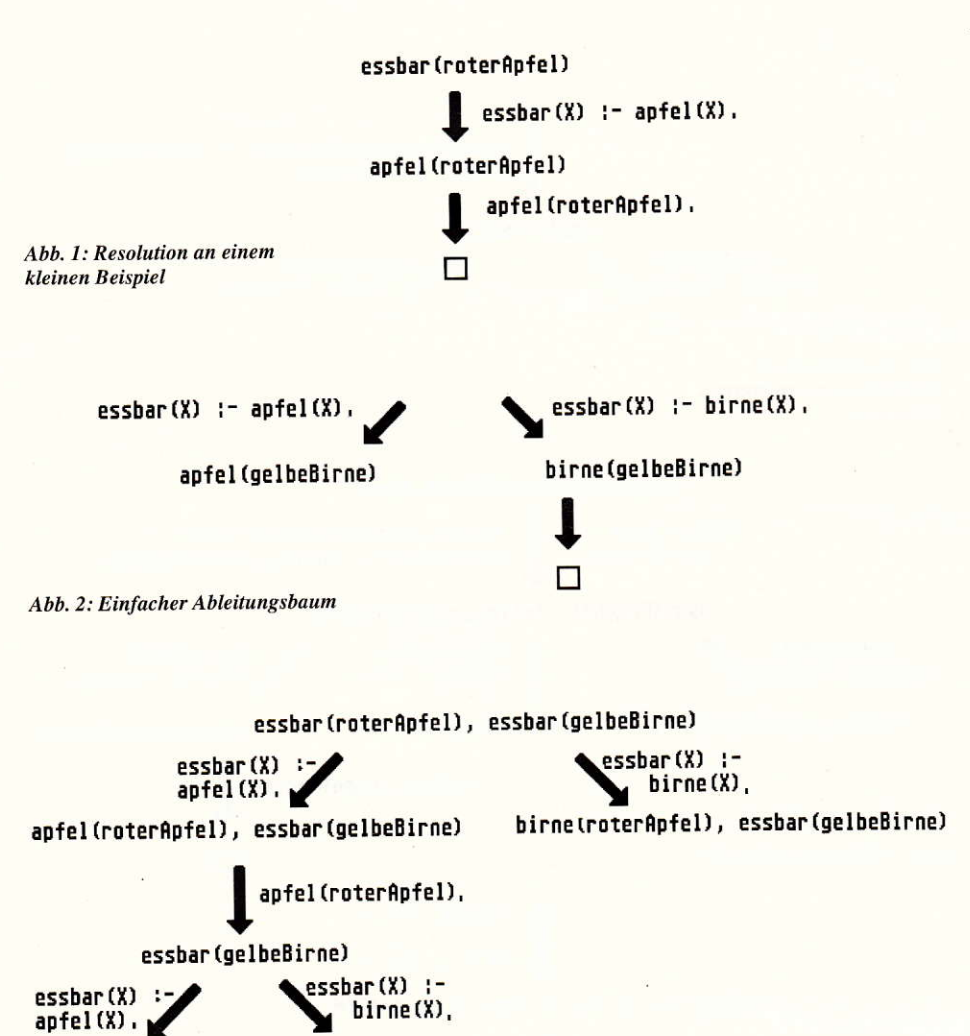


Abb. 1: Resolution an einem kleinen Beispiel

Abb. 2: Einfacher Ableitungsbaum

Abb. 3: Ableitungsbaum für eine Anfrage mit Konjunktion

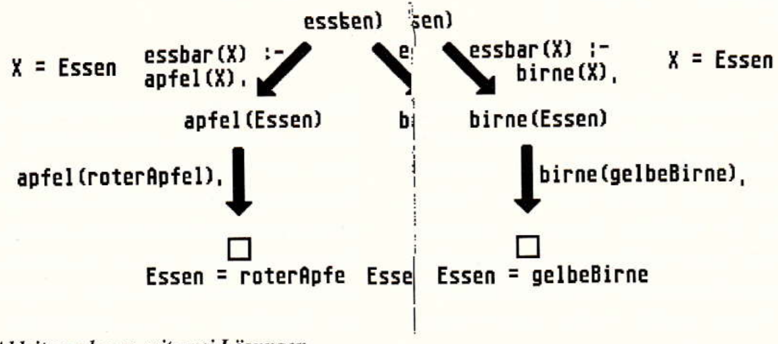


Abb. 4: Ableitungsbaum mit zwei Lösungen

In dem Apfelbeispiel paßt auf das Prädikat *essbar(roterApfel)* nur die Klausel *essbar(X) :- apfel(X)*, wobei die Variable *X* mit *roterApfel* gleichgesetzt wird. Nach Anwendung der Resolutionsregel wird aus *?- essbar(roterApfel)* also *?- apfel (roterApfel)*, d.h. die rechte Seite der Klausel (auch Klauselrumpf genannt) ersetzt das Prädikat, das auf die linke Seite der Klausel (Klauselkopf) paßt. Die durch einen Resolutionsschritt entstehende neue Anfrage wird als Resolvente bezeichnet. Das einzige Prädikat in der neuen Anfrage *?- apfel(roterApfel)* paßt offensichtlich auf das Fakt *apfel(roterApfel)*. Das Fakt hat keinen Klauselrumpf, und somit ist die Anfrage nach diesem Schritt leer. Eine solche leere Anfrage bedeutet das erfolgreiche Ende der Resolution. Tritt während der Resolution die Situation auf, daß die Anfrage noch nicht leer ist und auch kein Klauselkopf auf eines der Prädikate paßt, die in der Anfrage stehen, so schlägt die Resolution fehl. In Abb. 1 ist der komplette Ablauf noch einmal abgebildet. In der Abbildung wurde das Symbol „?-" vor den Anfragen weggelassen, und die leere Anfrage wird durch ein unausgefülltes kleines Quadrat symbolisiert. Die Pfeile kennzeichnen einen Resolutionsschritt und sind mit der Klausel beschriftet, die den jeweiligen Schritt ermöglicht. Eine Sequenz von Resolutionsschritten nennt man Ableitung. Eine Ableitung ist genau dann erfolgreich, wenn sie in der leeren Anfrage endet.

Die Qual der Wahl

Das Beispiel war natürlich recht einfach, da in jedem Schritt nur eine passende Klausel vorhanden war und die Anfrage immer aus einem Prädikat bestand. Sobald die Regeln etwas erweitert werden, ist das Vorgehen nicht mehr so eindeutig. Betrachten wir zum Beispiel die etwas erweiterte Obstauswahl:

```
essbar(X) :- apfel(X).
essbar(X) :- birne(X).
apfel(roterApfel).
birne(gelbeBirne).
```

Bei der Anfrage *?- essbar(gelbeBirne)* steht der Prolog-Interpreter vor der Qual der Wahl. Es paßt sowohl der Klauselkopf der Regel *essbar(X) :- apfel(X)* als auch der der Regel *essbar(X) :- birne(X)*. Die Lösung der Misere sind sogenannte Wahlpunkte (engl.: choice points). Sobald eine

Stelle erreicht wird, an der es mehrere passende Klauseln gibt, entsteht ein Wahlpunkt, und die textuell erste Klausel wird ausprobiert. Schlägt die Resolution im weiteren Verlauf fehl, wird zum letzten Wahlpunkt zurückgesetzt (engl.: backtracking) und die nächste Alternative versucht. In unserem Beispiel wird also zuerst die Klausel *essbar(X) :- apfel(X)* probiert, und somit heißt die Resolvente *?- apfel(gelbeBirne)*. Diese schlägt fehl, da es nur eine Klausel gibt, in der *apfel/1* im Kopf vorkommt, nämlich *apfel(roterApfel)*. Also wird zum letzten Wahlpunkt zurückgesetzt und dort die nächste passende Klausel probiert. Dies ist *essbar(X) :- birne(X)*, was zu der Resolvente *?- birne(gelbeBirne)* führt und somit schlußendlich die leere Anfrage zum Ergebnis hat, also gelingt.

In Abb. 2 sind die möglichen Ableitungen als Baum (Ableitungsbaum) dargestellt. Jeder Wahlpunkt führt in dem Anleitungsbaum zu einem Knoten, der zwei oder mehr Nachfolger besitzt. In Abb. 2 ist dies nur die Wurzel. Um das Ganze noch komplizierter zu machen, betrachten wir als nächstes eine Anfrage, die mehrere Prädikate enthält und erfragt, ob *roterApfel* und *gelbeBirne* eßbar sind:

```
?- essbar(roterApfel),
essbar(gelbeBirne).
```

In dieser Anfrage können beide Prädikate für einen Resolutionsschritt verwendet werden. Welche der beiden zuerst verwendet wird, ist auf den ersten Blick egal. Allerdings hat es, wie wir später noch sehen werden, Einfluß auf das Terminierungsverhalten. In Prolog wird grundsätzlich das am weitesten links stehende Prädikat einer Anfrage für den nächsten Resolutionsschritt verwendet. Die Beispielanfrage lautet somit nach dem ersten Resolutionsschritt:

```
?- apfel(roterApfel),
essbar(gelbeBirne).
```

Wieder wird das linke Prädikat zur Resolution benutzt, und die nächste Resolvente ist somit *?- essbar(gelbeBirne)*. Danach geht es wie gewohnt weiter. Der Ableitungsbaum zu dem Beispiel ist in Abb. 3 zu finden. Alle bisher gestellten Anfragen hatten nur eine Lösung, wie wir aber in der letzten Folge gesehen haben, kann Prolog durchaus mehrere Lösungen zu einer Anfrage liefern. Betrachten wir dazu die folgende Anfrage:

```
?- essbar(Essen).
```

Die Anfrage soll alle Belegungen der Variable *Essen* finden. Mit den einfachen Regeln des obigen Beispiels bleiben als Antworten nur *roterApfel* und *gelbeBirne*.

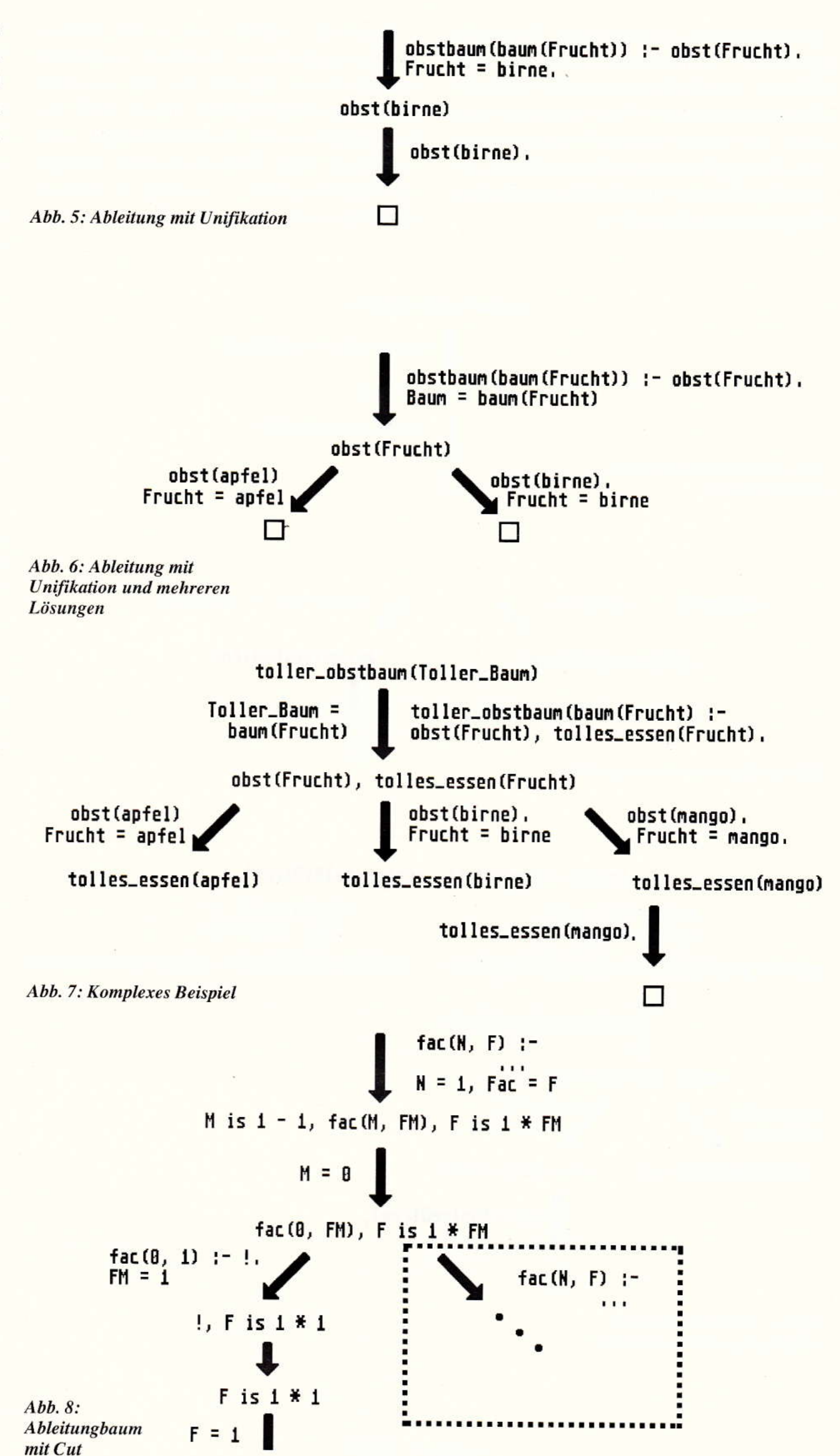


Abb. 5: Ableitung mit Unifikation

Abb. 6: Ableitung mit Unifikation und mehreren Lösungen

Abb. 7: Komplexes Beispiel

Abb. 8: Ableitungbaum mit Cut

Der Ableitungsbaum ist in Abb. 4 angegeben. Er besitzt zwei Ableitungen, die in der leeren Anfrage enden, und somit erfolgreich sind. Stellt man dem Prolog-System die Anfrage, liefert es zuerst die Antwort *Essen = roterApfel* und fragt dann, ob weitere Antworten erwünscht sind. Wird diese Frage bejaht (siehe letzte Folge), wird die zweite Antwort *Essen = gelbeBirne* geliefert. Die Reihenfolge der Antworten entspricht der textuellen Reihenfolge der Klauseln, zwischen denen eine Wahlmöglichkeit besteht, d.h. die weiter oben stehende Klausel wird zuerst gewählt. Um zu verstehen, wie die Variablen zu den verschiedenen Belegungen kommen und wie Prädikate, die kompliziertere Argumente besitzen, verarbeitet werden, müs-

sen wir uns genauer ansehen, wie Prolog feststellt, ob ein Prädikat auf einen Klauselkopf paßt.

Gleichgeschaltet

Der Mechanismus, der überprüft, ob ein Prädikat auf einen Klauselkopf paßt, und gleichzeitig dazu dient, Variablen mit Werten zu belegen, heißt Unifikation. Wird die Unifikation auf zwei Terme angewandt, so sagt man auch, daß die beiden unifiziert werden, d.h. die freien Variablen der Terme werden derart mit Werten belegt, daß die beiden Terme nach der Unifikation gleich sind. Da das Gleichheitszeichen in Prolog für die Unifikation steht, kann man deren Auswirkungen leicht ausprobieren. Versuchen wir zum Beispiel die folgende Anfrage:

```
?- c(X) = c(1).
```

Als Antwort liefert uns das Prolog-System *X = 1*, d.h. die Terme *c(X)* und *c(1)* sind unifizierbar, wobei die Variable *X* mit *1* belegt wird. Wird eine Variable mit einem Wert belegt, bedeutet das, daß alle Vorkommen der Variable (im Bindungsbereich) durch den Wert ersetzt werden. Man sagt auch, daß die Variable instanziert wird. Als nächstes probieren wir

```
?- c(X, X) = c(1, 1).
```

Die Antwort ist, wie erwartet, wieder *X = 1*. Anders verhält sich das System bei

```
?- c(X, X) = c(1, 2).
```

Hier lautet die Antwort *no*. Die beiden Terme sind also nicht unifizierbar, da die Variable *X* nicht gleichzeitig den Wert 1 und 2 annehmen kann. Man kann auch sagen, daß 1 nicht mit 2 unifizierbar ist. Eine wichtige Eigenschaft der Unifikation ist, daß beide Terme gleichwertig behandelt werden. Die folgenden beiden Anfragen sind also identisch:

```
?- c(X) = c(1).
?- c(1) = c(X).
```

Es können auch in beiden Termen gleichzeitig Variablen instanziert werden, wie dies im folgenden Beispiel der Fall ist:

```
?- c(1, Y) = c(X, 2).
```

Die Antwort lautet hier *X = 1, Y = 2*. Die Unifikation versucht also eine Belegung (Instanzierung) für die Variablen zu finden, die beide Terme gleichmacht. Ersetzt man die Variablen in den beiden Termen durch den Wert, der in der Antwort für die jeweilige Variable angegeben wird, so sind die beiden Terme gleich. Im letzten Beispiel würden beide Terme zu *c(1,2)* werden. Ein etwas komplizierteres Beispiel ist

Unifikation von zwei Termen T1 und T2

Fall 1: T1 besteht nur aus einer Variablen X

Instanziere X mit T2 und die Unifikation gelingt

Fall 2: T2 besteht nur aus einer Variablen X

Instanziere X mit T1 und die Unifikation gelingt

Fall 3: T1 und T2 sind zwei Zahlen

Falls die beiden Zahlen gleich sind, gelingt die Unifikation

Fall 4: T1 hat die Form p(A1, ..., An) und T2 hat die Form p(B1, ..., Bn)

Falls die Unifikation von A1 mit B1 und ... und An mit Bn gelingt, so gelingt die komplette Unifikation
Achtung: Die obige Schreibweise impliziert, daß beide Terme den gleichen Funktor und die gleiche Stelligkeit besitzen.

Fall 5: Sonst mißlingt die Unifikation

Tabelle 1: Unifikationsalgorithmus

```
?- node(leaf(1), node(X, leaf(3)))
= node(leaf(1), node(leaf(2),
leaf(Y))).
```

Das Prolog-System antwortet hierauf mit *X = leaf(2), Y = 3*. Der Term, der sich ergibt, wenn man die Werte der Variablen in die beiden unifizierten Terme einsetzt, ist *node(leaf(1), node(leaf(2), leaf(3)))*. Selbstverständlich können auch Variablen an Variablen oder Terme, die wieder Variablen enthalten, gebunden werden. Beipiele dafür sind etwa:

```
?- c(X) = c(Y).
?- c(X) = c(c(Y)).
```

Die Antwort ist hier *X = Y* bzw. *X = c(Y)*. Statt der Variablen *Y* kann in der Antwort auch so etwas wie *_273* stehen. Dies ist völlig gleichwertig, da die Namen von Variablen unbedeutend sind, solange ein Variablenname in allen Vorkommen durch denselben Namen ersetzt wird und nie zwei unterschiedlich benamte Variablen denselben Namen erhalten. Einen Term, der bis auf die Namen der Variablen gleich einem anderen Term ist, nennt man eine Kopie mit umbenannten Variablen.

Ein allgemeiner Unifikationsalgorithmus wird informell in Tabelle 1 beschrieben. In Fall 1 und 2 wird das Auftreten von freien Variablen behandelt, indem der andere Term einfach an die Variable gebunden wird. Dabei ist zu beachten, daß eine Variable, die schon an einen Wert gebunden ist, nicht als Variable, sondern wie der an sie gebundene Wert behandelt wird. In Fall 3 wird die Unifikation von Zahlen auf ihre Gleichheit zurückgeführt. Fall 4 behandelt die allgemeine Form eines Terms, der aus einem Atom mit oder ohne Argumentliste besteht. Solche Terme sind unifizierbar, sobald sie den gleichen Funktor mit der gleichen Stelligkeit besitzen und alle ihre Argumente paarweise unifizierbar sind.

Da die Unifikation als Hilfsmittel zur Resolution eingeführt wurde, soll jetzt deren Zusammenspiel beschrieben werden. Um eine Klausel während eines Resolutionsschrittes auszuwählen, wird untersucht, auf welche Klauselköpfe das in der Anfrage am weitesten links stehende Prädikat paßt. Wir können nun sagen, daß ein Prädikat genau dann auf einen Klauselkopf paßt, wenn die beiden unifizierbar sind. Die Variablen in der Klausel und im Prädikat werden durch die Unifikation mit Werten belegt. Sobald während der Resolution zu einem Wahlpunkt zurückgesetzt wird, werden alle Variableninstanzierungen, die zwischen dem Auftreten des Wahlpunkts und dem Fehlschlag durchgeführt wurden, wieder rückgängig gemacht.

Ein wichtiger Punkt ist dabei, daß die mehrfache Verwendung ein und derselben Klausel während der Resolution nicht mit denselben Variablen durchgeführt wird. Bevor der Klauselkopf mit dem Prädikat unifiziert wird, wird eine Kopie der gesamten Klausel (Kopf und Rumpf) gemacht, wobei alle Variablen umbenannt werden. Die Unifikation wird dann mit der Kopie durchgeführt, und falls diese gelingt, wird der Rumpf der Kopie in die Anfrage eingesetzt. Durch diesen Mechanismus entstehen die automatisch erzeugten Variablennamen, die aus einem Underscore und einer Zahl bestehen (z.B.: *_273*), und manchmal in Antworten des Prolog-Systems auftauchen.

Nehmen wir als erstes Beispiel die folgenden Klauseln:

```
obst(apfel).
obst(birne).
obstbaum(baum(Frucht)) :-
obst(Frucht).
```

Als Anfrage probieren wir

```
?- obstbaum(baum(birne)).
```

Auf das einzige Prädikat der Anfrage paßt nur der Kopf der dritten Klausel. Durch die Unifikation wird die Variable *Frucht* mit *birne* instanziert, und somit lautet die Resolvente: das Prädikat *obstbaum (baum(birne))* ersetzt durch *obst (Frucht)* mit instanzierter Variable) *?- obst(birne)*. Dieses Prädikat ist nur mit der zweiten Klausel unifizierbar und führt zur leeren

Anfrage und damit zum Erfolg. Die Ableitung dieses Beispiels ist in Abb. 5 dargestellt. Zusätzlich zur ausgewählten Klausel wird dort noch die Belegung der Variablen angegeben, die im zugehörigen Unifikationsschritt gebunden werden. Etwas komplizierter wird die Sache, wenn in der Anfrage schon eine Variable enthalten ist, wie etwa in

```
?-  obstbaum(Baum).
```

Die beiden Antworten sind *Baum = baum(apfel)* und *Baum = baum(birne)*. Wie das System auf diese Antworten kommt, ist in Abb. 6 dargestellt. Im ersten Schritt wird die Variable *Baum* mit *baum(Frucht)* instanziert. Im zweiten gibt es eine Wahlmöglichkeit zwischen den beiden Klauseln *obst(apfel)* und *obst (birne)*, die die Variable *Frucht* mit *apfel* bzw. *birne* instanzieren. Noch etwas komplexer ist das nächste Beispiel:

```
obst(apfel).
obst(birne).
obst(mango).
tolles_essen(mango).
tolles_essen(sushi).
toller_obstbaum(baum(Frucht)) :-
obst(Frucht), tolles_essen(Frucht).

?- toller_obstbaum(Toller_Baum)
```

Im ersten Resolutionsschritt wird, wie im vorherigen Beispiel, die Variable *Toller_Baum* mit *baum(Frucht)* instanziert. Allerdings lautet die Resolvente dieses Mal *?- obst(Frucht), tolles_essen(Frucht)*. Da nun zuerst das linke Prädikat bearbeitet wird und *obst(Frucht)* mit den Köpfen aller drei *obst/1*-Klauseln unifiziert werden kann, entsteht ein Wahlpunkt. Die textuell erste Klausel, also *obst(apfel)* wird als erstes gewählt und führt zur Instanzierung von *Frucht* mit *apfel* und zur Resolvente *?- tolles_essen(apfel)*. Dieses Prädikat ist aber mit keinem der vorhandenen Klauselköpfe unifizierbar und schlägt somit fehl. Da aber noch ein Wahlpunkt existiert, schlägt die komplette Anfrage noch nicht fehl, sondern es wird zum letzten Wahlpunkt zurückgesetzt.

Die nächste Wahl bringt die Belegung von *Frucht* mit *birne* und ist genauso erfolglos. Erst die dritte und letzte Wahl führt zu *Frucht = mango* und damit zur Resolvente *?- tolles_essen(mango)*. Deren Prädikat kann mit der vierten Klausel unifiziert werden und führt zum Erfolg der gesamten Anfrage mit der Antwort *Toller_Baum = baum(mango)*. Der Ableitungsbaum dieses Beispiels ist in Abb. 7 dargestellt.

Es ist wichtig zu bemerken, daß die Variableninstanzierung von *Frucht* bei jedem Fehlschlag rückgängig gemacht wird und somit bei der nächsten Wahl eine neue Instanzierung mit einem eventuell anderen Wert möglich ist. Die Instanzierung von *Toller_Baum* mit dem Wert *baum(Frucht)* wird hingegen nicht rückgängig gemacht, da sie vor dem Auftreten des Wahlpunkts geschah.

Nicht ist nicht gleich nicht

Als Abschluß der Beschreibung der Resolution soll noch eine Feinheit erläutert werden, deren Mißachtung schon zum Mißlingen so manchen Prolog-Programms geführt hat. Nachdem schon in der letzten Folge Konjunktionen und Disjunktionen eingeführt wurden, liegt es natürlich recht nahe, auch eine Negation (Verneinung) zu implementieren. Diese verhält sich allerdings etwas anders, als man auf den ersten Blick erwarten würde.

Wie eingangs erwähnt, basiert Prolog auf einer Einschränkung der Prädikatenlogik, die Hornlogik genannt wird. Für eine beliebige hornlogische Formel kann aber, aus hier nicht weiter ausgeführten Gründen, nicht unbedingt eine negierte Formel angegeben werden. In Prolog geht man deshalb einen anderen Weg. Soll festgestellt werden, ob die Negation einer Anfrage gilt, wird die Anfrage ausgeführt. Falls sie nicht zum Erfolg führt, wird angenommen, daß die Negation erfolgreich ist, anderenfalls nicht. Dies ist allerdings eine Vereinfachung der Dinge, da ein Mißerfolg des Prolog-Systems nicht wirklich die Unerfüllbarkeit einer Anfrage anzeigt, sondern nur, daß sie von Prolog im Rahmen der angegebenen Klauseln nicht abgeleitet werden kann. Zudem kann auch eine Nichtterminierung (endlose Rekursion) die Nichterfüllbarkeit einer Anfrage anzeigen. Sie führt mit der vorgestellten Methode der Negation aber nur zur Nichtterminierung der negierten Anfrage, statt deren Erfolg anzuzeigen. Gelingt eine negierte Anfrage, ist dies immer auf den Mißerfolg der nicht negierten Anfrage zurückzuführen. Deshalb werden beim Gelingen einer negierten Anfrage nie Variablen instanziert (schließlich gibt es ja auch keine Belegung, die zum Erfolg der nicht negierten Anfrage führt). Dies hat ein etwas mysteriöses Verhalten bei doppelter Negation zur Folge, da hier logischerweise erst recht keine Variablen instanziert werden, obwohl eine doppelte Verneinung streng genommen äquivalent zur nicht negierten Anfrage sein sollte. In Prolog wird eine Anfrage negiert, indem sie als Argument des Prädikats *not/1* angegeben wird. Ein Beispiel ist

```
?- not(1 = 3).
```

Prolog antwortet hier mit *yes*. Nach diesen etwas theoriebehafteten Erklärungen wenden wir uns nun etwas praxisorientierteren Problemen zu. Da Unifikation und Resolution für einige in der Praxis häufig auftretende Probleme nicht ausreichend sind, besitzt Prolog noch eine Reihe eingebauter Prädikate, die man sich zum großen Teil nicht selber programmieren kann oder die, in Prolog programmiert, zu langsam oder umständlich wären. Im folgenden werden wir einige dieser eingebauten Prädikate kennenlernen.

Wie in der ersten Folge schon erwähnt wurde, sind Terme in Prolog die grundlegende syntaktische Struktur. Selbst ein Prolog-Programm ist eine Reihe von aufeinanderfolgenden Termen. Dabei ist jede Regel oder Anfrage ein Term, der durch einen Punkt abgeschlossen wird. Die allgemeine Beschreibung eines Terms, die in der ersten Folge dieser Serie gegeben wurde, deckt aber keine Terme wie *p :- q* ab. In Prolog wird letzteres allerdings nur als alternative Schreibweise zu *:-(p, q)* oder *':-'(p, q)* angesehen. Um den Funktor eines zweistelligen Terms als Infixoperator, d.h. zwischen seinen beiden Argumenten, verwenden zu dürfen, muß man ihn als Infixoperator deklarieren. Dasselbe gilt für Prefix- und Postfixoperatoren, d.h. Operatoren, die einstellig sind und vor bzw. hinter ihrem Argument stehen. Alle drei Arten von Operatoren können mit Hilfe des Prädikats *op/3* deklariert werden.

Ein Beispiel ist in Listing 1 der ersten Folge zu finden. Dort wurde in der Zeile *:- op(200, yf, '!')*. das Ausrufezeichen als Postfixoperator mit einer Vorrangstufe von 200 deklariert (kann zwischen 1 und 1200 variieren). Die Angabe einer Regel ohne Kopf, die also mit „:-" beginnt, wird von Prolog ähnlich wie eine Anfrage behandelt. Das System versucht also *op(200, yf, '!')* zu erfüllen. Allerdings wird weder die Antwort *yes* bzw. *no* ausgegeben, noch wird eine Belegung für eventuell vorhandene freie Variablen angegeben. Daß ein Postfixoperator deklariert wird und dieser als Argument einen Term haben kann, dessen Funktor die gleiche Vorrangstufe besitzt, erkennt man an dem *yf*. Der Operator wird auch als schwach bezeichnet. Soll die Vorrangstufe im Argument immer kleiner sein, so schreibt man *xf* und nennt den Operator stark. Nach der Operatordeklaration sind Terme wie *5!* und *!(5)* gleichwertig. Auch die vordefinierten arithmetischen Operatoren wie „+", „-", „*" und „/" werden auf diese Weise als Operatoren deklariert.

Interessant ist, daß diese Operatordeklarationen nicht nur vom Prolog-System zum Einlesen von Programmen benutzt werden, sondern auch in Anwenderprogrammen zur Verfügung stehen. Das Standardprädikat zum Einlesen eines Terms ist *read/1*. Um es auszuprobieren, geben wir die folgende Anfrage ein:

```
?- read(X).
```

Danach erscheint keine Antwort vom Prolog-System, sondern es wartet auf die Eingabe eines Terms, der durch einen Punkt beendet wird. Geben wir *2 + 3 * 4.* ein und schließen die Eingabe durch ein Trennzeichen, also etwa ein Leerzeichen oder einen Zeilenvorschub (return), ab, so erhalten wir die Antwort *X = 2 + (3 * 4)*. Umgekehrt kann mit dem Prädikat *write/1* ein beliebiger Term ausgegeben werden, bei dessen Darstellung die Operatordeklarationen berücksichtigt werden. Ein Beispiel ist die Anfrage

```
?- write(+(1, 2)).
```

Sie führt zur Antwort *yes* und der gleichzeitigen Ausgabe von *1 + 2*. Eine Anfrage mit *write(1 + 2)* hätte dasselbe Ergebnis zur Folge gehabt. Im Gegensatz dazu gibt das Prädikat *display/1* einen Term aus, ohne daß die Operatordeklarationen beachtet werden. Ein kleines Beispielprogramm, das zum Experimentieren mit den Operatordeklarationen verwendet werden kann, ist in Listing 1 abgedruckt. In den ersten drei Zeilen werden drei verschiedene Operatoren deklariert. Der Infixoperator verhält sich derart, daß ein Ausdruck wie *1 ^ 2 ^ 3* implizit rechtsassoziativ geklammert wird, also als *1 ^ (2 ^ 3)*. Ein linksassoziativer Operator kann mit *yfx* deklariert werden und *xfx* erlaubt keine Operatoren gleicher Vorrangstufe neben sich. In der zweiten Zeile wird ein schwacher Postfixoperator deklariert. Er erlaubt Ausdrücke wie *3!!*. Im Gegensatz dazu ist etwas ähnliches mit dem starken Prefixoperator der dritten Zeile nicht möglich, d.h. „~~1" führt bei der Eingabe zu einer Fehlermeldung.

Das Prädikat *testops/0* liest zuerst einen Term ein und gibt ihn dann einmal mit und einmal ohne Berücksichtigung der Operatordeklarationen aus. Das zum Schluß stehende Prädikat *more/1* beendet die Rekursion und damit das Programm, falls das Atom *end* übergeben wird, sonst („_" steht für einen beliebigen Term) wird *testops/0* ein weiteres Mal ausgeführt.

Das Programm kann, wie in der letzten Folge beschrieben, mit *consult/1* werden und durch die Anfrage *?- testops.* ausgeführt werden. Durch Variieren der Operatordeklarationen (danach neues *consult/1* nicht vergessen) können verschiedene Operatoren und ihr Verhalten ausprobiert werden.

Ein mal eins

Wird als Argument eines Prädikats in Prolog der Term *1 * 1* angegeben, wird dieser nicht zu „1" ausgewertet, sondern er steht für den Term **(1, 1)* und wird von der Unifikation aus entsprechend behandelt, d.h. die folgende Anfrage liefert die Antwort *no*:

```
?- 1 * 1 = 1.
```

Um den Wert eines arithmetischen Ausdrucks zu berechnen, braucht man das eingebaute Prädikat *is/2*, das als Infixoperator deklariert ist. Die korrekte Anfrage ist also

```
?- 1 is 1 * 1.
```

Sie wird mit *yes* beantwortet. Auf diese Art können auch komplexere Ausdrücke berechnet werden, wie etwa in

```
?- X is 3 * 4 + 10 / 5.
```

Die Antwort lautet hier, wie erwartet, *X = 14*. Im Gegensatz zu Prädikaten wie *append/3* kann *is/2* mit freien Variablen im rechten Argument benutzt werden. Eine Anfrage wie *?- 5 is 3 + X* kann vom Prolog-System also nicht beantwortet werden.

Auch für die verschiedenen Relationen wie kleiner, kleiner oder gleich usw. sind in Prolog Operatoren vordefiniert. So gelingt etwa die Anfrage *?- 7 < 10.*, aber *?- 10 < 7.* gelingt nicht. Die Relationen besitzen die Operatoren „=", „\=", „<„ „=<„ „>", und „>=", wobei „=" und „\=" eigentlich für unifiziert und unifiziert nicht stehen, aber auch auf Zahlen angewendet werden können.

Einige weitere, oft benutzte Prädikate sind *var/1*, *nonvar/1*, *atom/1*, *integer/1* und *atomic/1*. Sie gelingen, sobald ihr Argument eine bestimmte Gestalt besitzt. Das Prädikat *var/1* gelingt, falls das Argument eine freie, also nicht instanzierte, Variable ist. Sobald sie instanziert ist, schlägt *var/1* fehl, dafür gelingt genau dann *nonvar/1*. Damit *atom/1* gelingt, muß das Argument ein Atom sein (dies kann natürlich auch eine Variable sein, die mit einem Atom instanziert ist). Entsprechend gelingt *integer/1* bei einer Zahl und *atomic*, falls das Argument ein Atom oder eine Zahl ist. Beispiele für diese Prädikate sind die Anfragen

```
?- var(X).
?- X = hallo, var(X).
?- X = hallo, atom(X).
```

Die erste Anfrage gelingt. Die zweite schlägt fehl, da *X* keine freie Variable mehr ist, sondern mit dem Atom *hallo* instanziert wurde. Dafür gelingt die dritte Anfrage wieder.

Schnipp, schnapp

Zum Abschluß wollen wir uns ein eingebautes Prädikat ansehen, mit dem die Anzahl der Lösungen, die durch die Resolution gefunden wird, eingeschränkt wird. Es heißt *Cut* und wird in Prolog durch ein Ausrufezeichen symbolisiert. Als Beispiel sehen wir uns das folgende Prädikat zur Berechnung der Fakultät an:

```
fac(0, 1).
fac(N, F) :-
        M is N - 1,
        fac(M, FM),
        F is FM * N.
```

Das Prädikat *fac/2* liefert im zweiten Argument die Fakultät der im ersten Argument stehenden Zahl. Die Anfrage *?-fac(3, F)* liefert also *F = 6*. Die Klauselauswahl für *fac(3, F)* ist eindeutig, da *fac(3, F)* nur mit dem Kopf der zweiten Klausel unifiziert werden kann. Dabei wird die Variable *N* mit „3" instanziert, und somit erhält *M* den Wert „2", und *fac/2* wird durch *fac(2, FM)* rekursiv angewendet. Die Rekursion endet, sobald das erste Argument „0" ist. In diesem Fall kann mit dem Kopf der ersten und der zweiten Klausel unifiziert werden. Da in Prolog zuerst die textuell erste Klausel probiert wird, endet die Rekursion, und als Ergebnis der Fakultät von „0" wird der Wert „1" geliefert. Allerdings wird auch ein Wahlpunkt erzeugt, da die zweite Klausel genauso anwendbar ist.

Bei einer Anfrage wie *?-fac(3, F)* bietet das Prolog-System nach der Ausgabe von *F = 6* die Möglichkeit, weitere Antworten zu erzeugen. Bei der Suche nach weiteren Antworten wird zum letzten Wahlpunkt zurückgesetzt und somit die zweite Klausel für ein Prädikat wie *fac(0, FM)* probiert. Dies führt aber zu einem rekursiven Aufruf mit *fac(-1, FM)*, der nicht terminiert und das Prolog-System somit in eine Endlosschleife führt. Eine Endlosschleife kann in MAXON-Prolog übrigens mit Shift-Shift-Help (beide Shift-Tasten) abgebrochen werden. Die Tatsache, daß *fac/2* auf der Suche nach weiteren Antworten in eine Endloschleife läuft, scheint beim direkten Aufruf in einer Anfrage nicht so schlimm. Bei der Benutzung innerhalb eines größeren Prolog-Programms kann es aber leicht zu fehlerhaften Programmen führen.

Eine naheliegende Methode, um die Endlosschleife zu verhindern, ist ein Test, der in der zweiten Klausel überprüft, ob *N* ungleich „0" ist. Das Prädikat lautet dann also

```
fac(0, 1).
fac(N, F) :-
        N \= 0,      % Test
        M is N - 1,
        fac(M, FM),
        F is FM * N.
```

Eine zweite, effizientere Möglichkeit, das Problem zu lösen, bietet der Cut. Er verhindert nicht nur, daß die zweite Klausel betreten wird, sondern löscht den Wahlpunkt, der dies ermöglichen würde, komplett. Durch das Löschen von Wahlpunkten, deren Alternativen sowieso nicht mehr gebraucht werden, spart das Prolog-System Zeit und Speicher, der durch das erfolglose Probieren dieser Alternativen und durch den Wahlpunkt selber verbraucht werden würde. Trotzdem sollte man im Umgang mit dem Cut immer etwas Vorsicht walten lassen, da man auch leicht mal einen Wahlpunkt löscht, der zu wichtigen Lösungen geführt hätte. Doch nun *fac/2* mit Cut:

```
fac(0, 1) :- !.      % Cut
fac(N, F) :-
        M is N - 1,
        fac(M, FM),
        F is FM * N.
```

Sobald die erste Klausel ausgewählt wird und das Prolog-System auf den Cut trifft, werden alle Wahlpunkte gelöscht, die erzeugt wurden, seitdem die Klausel ausgewählt wurde, in der der Cut steht (inklusive dem Wahlpunkt, den die Auswahl dieser Klausel eventuell erzeugt hat). In unserem Beispiel ist das nur der Wahlpunkt, der bei der Klauselauswahl für *fac(0, FM)* erzeugt wurde. Stehen im Klauselrumpf links von dem Cut weitere Prädikate, werden auch die Wahlpunkte gelöscht, die durch die Resolution dieser Prädikate erzeugt wurden. In Abbildung 8 ist die Wirkung des Cuts anhand des Ableitungsbaums für *fac(1, F)* dargestellt. Der gestrichelte Kasten enthält den Teil des Baums, der durch den Cut abgeschnitten, d.h. der durch das Löschen des Wahlpunktes nicht ausprobiert wird.

Ein weiteres Beispiel für die Verwendung des Cuts ist in Listing 2 abgebildet. Dort ist das Prädikat *testops/0*, das schon in Listing 1 vorgestellt wurde, ohne Rekursion realisiert. Die Arbeit wird in dieser Version von einem Hilfsprädikat namens *testopsLoop/0* gemacht. In diesem wird das eingebaute Prädikat *repeat/0* benutzt. Es gelingt immer und kann beim Rücksetzen beliebig oft wiedererfüllt werden. Es ist wie folgt definiert:

```
repeat.
repeat :- repeat.
```

In den drei darauffolgenden Zeilen wird ein Term eingelesen und wie in der ersten Version des Prädikats ausgegeben. Ist der

```
:- op(400, xfy, '^'),   % linksassoziativer Infixoperator
   op(200, yf, '!'),    % Postfixoperator (schwach)
   op(200, fx, '~').    % Prefixoperator (stark)

testops :-
        read(Term),
        nl, write(Term),
        nl, display(Term), nl,
        more(Term).

more(end).
more(_) :- testops.
```

Listing 1: Operatortestprogramm

```
:- op(400, xfy, '^'),   % linksassoziativer Infixoperator
   op(200, yf, '!'),    % Postfixoperator (schwach)
   op(200, fx, '~').    % Prefixoperator (stark)

testops :- testopsLoop.
testops.

testopsLoop :-
        repeat,
        read(Term),
        nl, write(Term),
        nl, display(Term), nl,
        Term = end,
        !, fail.
```

Listing 2: Operatortestprogramm ohne Rekursion

eingegebene Term ungleich *end*, scheitert *Term = end*, und es wird bis zu *repeat* zurückgesetzt.

Da *repeat/0* immer einen weiteren Wahlpunkt besitzt, werden die darauffolgenden Prädikate noch einmal abgearbeitet. Interessant wird es, sobald der Term *end* eingegeben wird. Dann gelingt *Term = end*, und es werden der Cut und das Prädikat *fail/0* abgearbeitet. Der Cut löscht alle Wahlpunkte, also auch den von *repeat/0*, seit dem Beginn der Abarbeitung von *testopsLoop/0*. Das Prädikat *fail/0* schlägt immer fehl und sorgt so insgesamt für ein Fehlschlagen von *testopsLoop/0*. Damit das Prolog-System bei einer Anfrage wie *?- testops.* als Antwort *yes* und nicht *no* liefert, ist die zweite Klausel des Prädikats *testops/0* da. Sie sorgt für ein Gelingen der gesamten Anfrage, obwohl die erste Regel, nach der Eingabe von „end", fehlschlägt.

Obwohl man die Funktion eines Prädikats wie *fac/2* ohne weiteres mit der Resolution erklären kann, ist eine an imperative Sprachen angelehnte Erklärung manchmal verständlicher. Betrachtet man jedes Prädikat als Prozedur, wobei ein Prädikat erst eindeutig durch die Kombination von Funktor und Stelligkeit bestimmt ist, kann man das Auftauchen eines Prädikats in einer Anfrage oder dem Rumpf einer Regel als Prozeduraufruf ansehen. Freie Variablen in einem solchen Prozeduraufruf stehen meist für Ausgabeparameter, und freie Variablen im Kopf einer Regel dienen als Eingabeparameter.

Im Gegensatz zur imperativen Programmierweise ist es in Prolog allerdings oftmals möglich, denselben Parameter sowohl zur Ein- als auch zur Ausgabe zu verwenden, wie wir in der letzten Folge etwa an *append/3* gesehen haben. Wenn wir das weiter oben vorgestellte Prädikat *fac/2* auf diese Weise betrachten, stellt die Anfrage *?- fac(5, F)* einen (Prozedur-) Aufruf von *fac/2* dar. Eingabeparameter ist das erste Argument, in dem die Zahl „5" übergeben wird. Als Ausgabeparameter dient das zweite Argument, hier also die Variable *F*. Das Prädikat bzw. die Prozedur *fac/2* ist derart formuliert, daß im Fall eines Eingabewerts von „0" der Wert „1" zurückgegeben wird. Sonst wird von der Eingabe eins abgezogen und *fac/2* mit dem Ergebnis der Subtraktion aufgerufen. Das Ergebnis dieses rekursiven Aufrufs bildet, multipliziert mit dem Eingabewert, das Resultat. Diese prozedurale Sichtweise ist für Programme oder Programmteile, die viel Ein-/Ausgabe oder Arithmetik ausführen, oft übersichtlicher und verständlicher als eine Interpretation mittels Resolution.

So weit, so gut

In dieser Folge haben wir die Funktionsweise von Prolog kennengelernt. Das nächste Mal sehen wir, wie man mit den Mitteln, die in dieser Folge vorgestellt wurden, recht schnell und komfortabel Programme schreiben kann. Dabei werden wir uns sehr stark mit der Behandlung von Datenstrukturen beschäftigen, die in vielen Programmen vorteilhaft eingesetzt werden können.

Manuel Chakravarty

Literatur:
[1] Clocksin/Mellish: „Programming in Prolog", Springer



ENVIRONMENT STRINGS

Teil 2: Anwendungen unter TOS

Nachdem im ersten Teil dieser Serie die Grundlagen erläutert wurden, möchte ich nun einige Anwendungen zeigen. Jedes Programm, das weitere nachlädt, sollte die Directories der Variablen PATH nach dem zu ladenden Programm durchsuchen. Pfade werden in Environment-Variablen durch Komma oder Semikolon getrennt aufgelistet, das aktuelle Directory ist '.', ein abschließendes '\' ist nicht erforderlich.

Ein Beispiel für die Suche in PATH ist die Funktion findfile. Sie sucht file in den Verzeichnissen von PATH. Ist PATH nicht vorhanden, wird im aktuellen Verzeichnis gesucht. Als Demo für diese Funktion dient *FF_DEMO*. Dieses Programm sucht *ENV.TOS* (aus Teil 1). *ENV.TOS* kann nun in einem der Verzeichnisse von PATH versteckt werden, *FF_DEMO* wird es schon finden.

Eine weitere standardisierte Anwendung für Environment-Variablen ist das XARG-Verfahren, das auf Allan Pratt und Dale Schumacher zurückgeht. Es erlaubt die Übergabe von beliebig vielen Parametern an Programme. Die Standard-Kommandozeile ist ja auf 124 Zeichen begrenzt. Man legt folgende Struktur an:

```
typedef struct
{
char xarg_magic[4];  /* enthält
                        "xArg" */
int xargc;           /* entspricht
                        argc in main() */
char **xargv;        /* entspricht argv
                        in main() */
char *xiovector;       /* z.Zt.
                        unbenutzt */
char *xparent;       /* Zeiger auf Base-
                        page des aufru-
                        fenden Programms
                     */
} XARG;
```

Einen Zeiger auf diese Struktur übergibt man in der Environment-Variablen *xArg*, und zwar als achtstellige Hex-Konstante, etwa „xArg=000310D8". Das aufgerufene Programm überprüft zunächst die Gültigkeit der Struktur. *Xarg_magic* muß „xArg" enthalten, *xparent* muß auf die Basepage des aufrufenden Programmes zeigen (dieser Zeiger ist in der eigenen Basepage enthalten). Dann sind die Argumente unbedingt zu kopieren, denn der Speicher gehört dem aufrufenden Programm.

Ein ähnliches Verfahren mit der Bezeichnung *ARGV* wurde vor einiger Zeit von Atari USA zum Standard erhoben. Dabei wird die gesamte Kommandozeile im Environment übergeben. Der Variablen *„ARGV="*, deren Inhalt beliebig ist, folgen alle zu übergebenden Argumente, jeweils durch 0 getrennt. Das erste Byte der normalen Kommandozeile, das die Länge enthält, wird auf 127 gesetzt (normalerweise <= 124). Das aufgerufene Programm muß die ganze Sache aus dem Environment löschen. Dazu überschreibt es das 'A' von „ARGV" mit 0. Der Vorteil gegenüber XARG ist, daß sich alle Parameter im Speicher des nachgeladenen Programms befinden. Das Environment wird ja vom TOS kopiert.

Hierzu das Programm *STARTUP.TTP*. Es verarbeitet XARG- und ARGV-Argumente und kann als startup für C-Programme benutzt werden. Als Demo dient xargdemo.tos, das seine Parameter an startup.ttp per Kommandozeile, per XARG und per ARGV übergibt.

Jan Bolt

Literatur:
[1] Atari ST Profibuch, Jankowski, Reschke, Rabich, Sybex 1988

```
 1:  /* ============================================*
 2:   *                                             *
 3:   * findfile.c                                  *
 4:   *                                             *
 5:   * Sucht file in allen Verzeichnissen von PATH, *
 6:   * gibt kompletten Pfadnamen zurück,           *
 7:   * oder NULL bei vergeblicher Suche            *
 8:   *                                             *
 9:   * 24.05.90 Jan Bolt                           *
10:   *                                             *
11:   * Turbo C                                     *
12:   * (c) MAXON Computer GmbH 1991 ===============*/
13:
14:  #include <stdio.h>
15:  #include <stdlib.h>
16:  #include <string.h>
17:  #include <tos.h>
18:
19:  #define is_file(f) (Fsfirst(f,0) == 0)
20:
```

```
21:  char *findfile(char *file)
22:       {
23:       static char tmp[PATH_MAX];
24:       char *path, *p;
25:
26:       if (strchr(file, '\\') || strchr(file,
              ':')) /* file enthält Pfad */
27:          return (is_file(file)) ? file : NULL;
28:
29:       if ((path=getenv("PATH")) == NULL ||
             /* falls PATH nicht existiert */
30:           *path == '\0')  /* oder leer */
31:          path = ".";  /* path = akt Dir */
32:
33:       while (*path != '\0')
34:             {
35:             p = tmp;
36:             while (*path != '\0' &&
37:                    *path != ';'  &&
38:                    *path != ',')
```

→

```
                        /* Directory aus PATH */
39:              *p++ = *path++;
                        /* nach tmp kopieren  */
40:         if (*path != '\0')
                   /* Separator überspringen */
41:            path++;
42:         if (p[-1] != '\\')
                        /* Directory mit \ */
43:            *p++ = '\\';    /* abschließen */
44:         strcpy(p, file);          /* file
             anhängen */
45:         if (is_file(tmp))
46:            return tmp;
47:         }
48:      return NULL;
```

```
 1: /* ============================================*
 2:  *                                            *
 3:  * ff_demo.c                                  *
 4:  *                                            *
 5:  * Demo findfile                              *
 6:  *                                            *
 7:  * muß mit putenv.o, findfile.o gelinkt werden *
 8:  *                                            *
 9:  * 24.05.90 Jan Bolt                          *
10:  *                                            *
11:  * Turbo C                                    *
12:  * (c) MAXON Computer GmbH 1991 ==============*/
13:
14: #include <stdio.h>
15: #include <stdlib.h>
16:
17: /*=============== Hauptprogramm ==============*/
18:
19: int main()
20:     {
21:     static char file[] = "env.tos";
22:     char *p, *findfile(char *file);
23:     int putenv(char *entry);
24:
25:     if (getenv("PATH") == NULL)
26:        putenv("PATH=\\bin;bin;.");
27:
28:     if ((p = findfile(file)) == NULL)
29:        printf("%s nicht gefunden !\n", file);
30:     else
31:        printf("%s\n", p);
32:
33:     getchar();
34:
35:     return 0;
36:     }
```

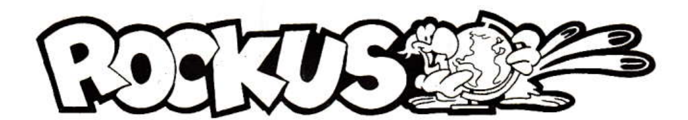
```
 1: /*============================================*
 2:  *                                            *
 3:  * startup.c                                  *
 4:  *                                            *
 5:  * Startup Modul für Turbo C                  *
 6:  *                                            *
 7:  * - Verarbeitung von XARG Argumenten         *
 8:  * - Verarbeitung von ARGV Argumenten         *
 9:  *                                            *
10:  * 07.06.90 Jan Bolt   Version 210990         *
11:  *                                            *
12:  * Turbo C                                    *
13:  * (c) MAXON Computer GmbH 1991 ==============*/
14:
15: #include <stdio.h>
16: #include <stdlib.h>
17: #include <string.h>
18: #include <tos.h>
19:
20: /*======== Hauptprogramm ====================*/
21:
22: int do_main(int argc, char *argv[])
23:     {
24:     int i;
25:
26:     for (i = 0; i < argc; i++)
27:         printf("argv[%d]='%s'\n", i, argv[i]);
28:
```

```
29:     return 0;
30:     }
31:
32: /* strdup() fehlt in TC Lib */
33:
34: char *strdup(const char *str)
35:     {
36:     char *p;
37:
38:     if ((p = malloc(strlen(str)+1)) != NULL)
39:        strcpy(p, str);
40:     return p;
41:     }
42:
43: /*====== XARG-, ARGV- Verarbeitung ==========*/
44:
45: #define ENSMEM -39           /* TOS-Fehler out
                                    of mem */
46: #define XARG_MAGIC 0x78417267 /* "xArg" */
47:
48: typedef struct               /* XARG-Struktur */
49:         {
50:         char   xarg_magic[4];
51:         int    xargc;
52:         char   **xargv;
53:         char   *xiovector;
54:         BASPAG *xparent;
55:         } XARG;
56:
57: int main(int nargc, char *nargv[])
58:     {
59:     extern BASPAG *_BasPag;
60:     XARG *xarg;
61:     int argc, i, rv;
62:     char *xenv, **argv;
63:
64:     if ((xenv = getenv("xArg")) != NULL &&
65:        (xarg = (XARG *)strtoul(xenv, NULL, 16))
           != NULL &&
66:        ((long)xarg & 1) == 0 &&
67:        *(long *)xarg->xarg_magic==XARG_MAGIC &&
68:        xarg->xparent == _BasPag->p_parent)
69:       {
70:       printf("XARG:\n");
71:       /* XARG-Struktur ist gültig. Argumente
             müssen kopiert */
72:       /* werden, sie befinden sich im Speicher
             des Parent.  */
73:       /* Speicher wird dynamisch mit malloc
             zugewiesen      */
74:       argc = xarg->xargc;
75:       if ((argv = malloc((argc+1)*sizeof(char *
              ))) == NULL)
76:          return ENSMEM;
77:       argv[argc] = NULL;
78:       for (i = 0; i < argc; i++)
79:           if ((argv[i] = strdup(xarg-
                 >xargv[i])) == NULL)
80:              return ENSMEM;
81:
82:       rv = do_main(argc, argv);
83:
84:       for (i = 0; i < argc; i++)
85:           free(argv[i]);
86:       free(argv);
87:
88:       return rv;
89:       }
90:
91:     if ((xenv = getenv("ARGV")) != NULL &&
92:       *_BasPag->p_cmdlin == 127) /* Länge
                         Kommandozeile = 127? */
93:       {
94:       printf("ARGV:\n");
95:       /* ARGV ist gültig. Es gilt einen Vektor
             von      */
96:       /* Zeigern auf die Argumente zu bilden,
             argc      */
97:       /* ist zu diesem Zeitpunkt leider nicht
             bekannt.  */
98:       /* Speicher wird dynamisch mit malloc
             zugewiesen. */
99:       xenv[-5] = '\0';  /* "ARGV=" löschen */
100:      while (*xenv++)    /* ggf Wert
                              überspringen */
101:         ;
102:      argc = 0;
```

→

```
103:        if ((argv = malloc(sizeof(char *))) ==
                NULL)
104:            return ENSMEM;
105:        while (*xenv)
106:            {
107:            argv[argc++] = xenv;
108:            if ((argv = realloc(argv, (argc+1)*
                    sizeof(char *))) == NULL)
109:                return ENSMEM;
110:            while (*xenv++)
111:                ;
112:            }
113:        argv[argc] = NULL;
114:
115:        rv = do_main(argc, argv);
116:
117:        free(argv);
118:
119:        return rv;
120:        }
121:
122:    printf("Kommandozeile:\n");
123:    return do_main(nargc, nargv);
124:    }
```

```
  1: /*==============================================*
  2:  *                                              *
  3:  * xargdemo.c                                   *
  4:  *                                              *
  5:  * Demo XARG-Verfahren                          *
  6:  *                                              *
  7:  * muß mit putenv.o gelinkt werden              *
  8:  *                                              *
  9:  * 17.02.90 Jan Bolt  Version 210990            *
 10:  *                                              *
 11:  * Turbo C                                      *
 12:  * (c) MAXON Computer GmbH 1991 ===============*/
 13:
 14: #include <stdio.h>
 15: #include <stdlib.h>
 16: #include <tos.h>
 17:
 18: typedef struct
 19:         {
 20:         char     xarg_magic[4];  /* enthält
                                        "xArg" */
 21:         int      xargc;          /* argc */
 22:         char     **xargv;        /* argv */
 23:         char     *xiovector;     /* bisher
                                        unbenutzt */
 24:         BASPAG   *xparent;       /* Zeiger auf
            aufrufendes Prg */
 25:         } XARG;
 26:
 27: /*=============== Hauptprogramm ===============*/
 28:
 29: main()
 30:     {
 31:     int putenv(char *entry);
 32:     extern BASPAG *_BasPag;   /* Zeiger auf
                                    eigene Basepage */
 33:     char env[14], *p;
 34:     static char cmd[] = "\x8p1 p2 p3";
 35:     static char *xargs[] = {"startup",
 36:                             "xp1",
 37:                             "xp2",
 38:                             "xp3",
 39:                             NULL};
 40:
 41:     static XARG xarg = {'x','A','r','g',4,xargs,
                            NULL,NULL};
 42:
 43:     xarg.xparent = _BasPag;
 44:
 45:     /* Parameter per Kommandozeile */
 46:     Pexec(0, "startup.ttp", (COMMAND *)cmd,
             NULL);
 47:     getchar();
 48:
 49:     /* Parameter per XARG */
 50:     sprintf(env, "xArg=%08lX", &xarg); /* Zeiger
                                       auf xarg in */
 51:     if (!putenv(env))     /* in xArg=übergeben */
 52:         fprintf(stderr, "environment overflow
                 !\n");
 53:     else
 54:         Pexec(0, "startup.ttp", (COMMAND *)cmd,
                 NULL);
 55:     getchar();
 56:     putenv("xArg");          /* xArg löschen */
 57:
 58:     /* Parameter per ARGV */
 59:     if (!putenv("ARGV= startup ap1 ap2 ap3"))
 60:         fprintf(stderr, "environment overflow
                 !\n");
 61:     else
 62:         {
 63:         /* Argumente durch 0 trennen */
 64:         p = getenv("ARGV");
 65:         while (*p)
 66:             if (*p++ == ' ')
 67:                 p[-1] = 0;
 68:         cmd[0] = 127;
 69:         Pexec(0, "startup.ttp", (COMMAND *)cmd,
                 NULL);
 70:         }
 71:     getchar();
 72:
 73:     return 0;
 74:     }
```







die Möglichkeit ihrer Änderung - Optimierung oder Erweiterung - offenzuhalten.

Wenn man also über die Kenntnis des leider nur strukturmäßig wohldokumentierten Ankers, eben des MPB, Zugriff auf diese ebenfalls in ihrer Existenz und Struktur dokumentierten einfach verketteten Listen hätte, könnte man die Speicherverwaltung des GEMDOS allerliebst manipulieren. Dazu eröffnen sich folgende Perspektiven:

Ein süßer Traum...

Atari dokumentiert endlich die Adresse des MPB als Systemvariable und gibt damit den Zugriff auf die schon längst publizierten MDs frei. Dafür sollte man Allan Pratt und seinem Team von Systemprogrammierern den 'Goldenen Jack am Band' verleihen! Gewisse 'trickreiche' Programme könnten dann mit Tabellen für Abfragen erster Art versehen werden und endlich in den Bereich der so lang ersehnten programmiertechnischen Legalität überwechseln.

...und die harte Wirklichkeit

Es gibt da die allgemein bekannte, dokumentierte [2] und vom Anwender tunlichst nicht aufzurufende BIOS-Funktion Nr.0 *getmpb*, die vom GEMDOS während des Systemstarts dazu benutzt wird, den MPB auszufüllen und dabei gleichzeitig einen MD, nämlich **den** MD, das ist die Systemvariable *themd* $48E, zu initialisieren [2, 4].

Da wir wissen, daß das GEMDOS nur einmal die Funktion BIOS (0) *getmpb* aufruft, müssen wir uns zu einem Zeitpunkt in der Systeminitialisierung, zu dem der BIOS-Trap schon eingerichtet ist, aber vor der Initialisierung des GEMDOS, in den Trap hängen und warten, bis *getmpb* aufgerufen wird. Dabei wird *p_mpb* als Zeiger auf den MPB auf dem Stack übergeben, wir brauchen ihn uns nur zu merken und können ihn in einer dokumentierten und deshalb für andere Programme zugänglichen Form ablegen, z.B. als *cookie* im *cookie jar*, um gleich einmal diese mit TOS 1.6 eingeführte, aber auch bei früheren TOS-Versionen nachrüstbare Informationsstruktur zu nützen.

Leider stellt sich bei Untersuchung der im schon zitierten HHG [2] sehr ausführlich dokumentierten Startup-Sequenz heraus, das nur eine ROM-Port-Cartridge dort hineinkommen kann, und zwar in unserem Falle am besten eine Anwendung vom Typ 1, direkt vor der Initialisierung des GEMDOS. Nun haben aber Cartridges, auch ROM-Module genannt, einen Nachteil: Sie sind ziemlich hart und lassen sich weder auf Diskette ziehen noch über ein elektronisches Medium schicken, was ihrer Verbreitung natürlich im Wege steht - zumal dies ja eine der wenigen echten ROM-Modul-Anwendungen wäre. Na, zumindest den Code kann man ja veröffentlichen. Leider ist diese harte Tour bis dato der einzige Weg, dem MPB auf völlig legale - und damit 100%ig und für alle Zeiten (...solange das TOS noch von Belang ist und Atari sich an seine eigene Dokumentation hält...) sichere - Weise beizukommen.

Weichere Möglichkeiten

Der Anker der MD-Listen ist also der MPB. Nur, was macht man, wenn man seine Adresse nicht kennt? Das einzige, worauf man seine Hand legen kann, ist *themd* ($48E), eine 'garantierte' Systemvariable zwar, aber leider nicht sehr von Nutzen. Das war und ist der bisherige Stand der (recht kärglichen) Dokumentationen und die generelle Meinung der Fachautoren. Das soll sich nun ändern, denn genau hier setze ich den Hebel an!

themd ist gleich nach der GEMDOS-Initialisierung ein (der einzige) Eintrag der *mfl* und weist damit den gesamten Anwenderspeicher - auch TPA (Transient Program Area) genannt - von *membot* bis *memtop* als frei aus, so ist's dokumentiert in [2]. Die *mal* ist zu diesem Zeitpunkt leer (mp_mal=0), *mp_mfl* (und bis TOS 1.6 auch mp_rover) zeigt auf themd. Dies ändert sich, sobald die erste Speicherreservierung (z.B. zum Starten eines Prozesses) vorgenommen wird. Nun ist *themd* der letzte MD der *mal*: *m_link*=0, *m_own* = Urprozeß (seit TOS 1.4, früher Null), *m_start* = *membot*, *m_length* enthält die Größe des ersten reservierten Blocks. Die Möglichkeit zu einer solchen 'öffentlichen' Reservierung - mit *Malloc()* - besteht bei ausführbaren Boot-Sektoren (Floppy- und DMA-Boot) und Päckchen, in dieser Reihenfolge. Danach ist dann wieder das ROM dran, und hier geschehen auf jeden Fall die ersten Reservierungen für Environment-String und Basepage, die für den sog. AUTOEXEC-Prozeß eingerichtet werden (nicht dokumentiert), der dann, falls vorhanden, die AUTO-Ordner-Programme und anschließend das AES (oder das COMMAND.PRG) startet (das ist wieder dokumentiert!).

Fallunterscheidung

1. Wir haben in *themd* eine Struktur, auf die zu einem Zeitpunkt, zu dem noch kein Speicher reserviert wurde (Boot-Sektorprogramme und Päckchen), ein oder zwei Zeiger (*mp_mfl* und *mp_rover*) zeigen. Die müßten sich doch finden lassen, und damit wäre dann der ach so heißbegehrte MPB lokalisiert [5].

Äußerst kritisch ist hier, daß noch kein Speicher reserviert worden sein darf, im Falle des Päckchens könnte z.B. ein Harddisk-Treiber aus dem Boot-Sektor von Laufwerk C: dazwischen gekommen sein, so daß eine gewisse Sicherheit für diese Methode nur dann besteht, wenn das Programm aus dem Boot-Sektor von Laufwerk A: startet und damit garantiert als allererstes ausgeführt wird. Die Unberührtheit des Speichers läßt sich recht einfach nachprüfen. Um dem Problem mit der Benutzung des *mp_rover* aus dem Wege zu gehen, empfiehlt sich hier die Suche von unten nach oben. Die verbleibende Unsicherheit resultiert aus der nicht ganz dokumentenechten Annahme, daß es im durchsuchten Bereich nur einen solchen Zeiger gibt, und daß das dann der gesuchte *mp_mfl* ist (ich wüßte allerdings nicht, was das GEMDOS mit mehr als einem solchen Zeiger auf die *mfl* anfangen sollte!).

2. Wir haben in *themd* das letzte Glied einer Kette, auf deren Beginn *mp_mal* zeigt. Dies ist zu allen späteren Zeiten so, gleich, ob aus dem AUTO-Ordner oder vom Desktop aus. Dieser Fall ist etwas kniffliger: Zeiger suchen, der auf *themd* zeigt; prüfen, ob's ein *m_link* eines MD sein könnte; prüfen, ob *m_own* Null ist; wenn ja, ist es vermutlich der *mp_mal*; sonst: Zeiger suchen, der auf diesen MD zeigt...

Hier gibt es zwei kritische Punkte, den einen allerdings erst seit TOS 1.4. Die von residenten Programmen belegten MDs werden nämlich wie schon zuvor üblich aus der *mal* ausgehängt und sind damit dem GEMDOS unbekannt. [6] Leider werden die von ihnen belegten 'Slots' aber nicht freigegeben, d.h. der Inhalt gelöscht oder zumindest ihr *m_own* ungültig gemacht. Bei z.B. 10 residenten AUTO-Ordner-Programmen sind dann 20 MD-Slots völlig unnötigerweise belegt (je 10 für die Environment-Strings, 10 für die residenten Programmteile inklusive Basepage), ein weiterer Bug im neuen GEMDOS-Poolmanager (POOLFIX3 hilft da leider auch nicht weiter)! Unser Suchprogramm könnte somit auf die Adressen zwar formal gültiger, aber für uns wertloser MDs stoßen, die als 'Leichen' im GEMDOS-Pool liegen. Ein Test auf die Gültigkeit von *m_own* (muß auf gültige Basepage zeigen [7]) nützt in diesem Falle auch

nichts. Der Referenzierungsalgorithmus, der die verkettete Liste von hinten aufrollen soll, muß also durch einen entsprechenden Dereferenzierungsalgorithmus ergänzt werden, damit sich die Routine sozusagen wieder rückwärts aus der Sackgasse - Stufe für Stufe - heraushangeln kann, um dann jeweils neue Versuche nach vorn zu starten. Damit das Programm bei etwaigem Versagen nicht hängenbleibt, sollte die maximal mögliche Anzahl der 'Backtrakking'-Versuche auf einen vernünftigen Wert begrenzt werden. Das alles hört sich zwar furchtbar kompliziert an, der Code ist jedoch recht kurz und klar!

Der zweite kritische Punkt ist die Abbruchbedingung, nämlich *m_own*=0. So etwas darf es allerdings in der *mal* nicht geben: einen herrenlosen Block. Dann hätten wir also hoffentlich den *mp_mal* selbst erwischt. Das wäre dann aber kein Element eines MD mehr, und die Tatsache, daß an diesem Offset eine Null steht, haben wir nur dem glücklichen Umstand zu verdanken, daß der MPB (zumindest zur Zeit des AUTO-Ordners und meist auch des Desktops) allein auf weiter Flur in einem sonst leeren Datenbereich steht. Dies ist natürlich logisch überhaupt nicht zwingend. Und das muß auch nicht so bleiben, selbst wenn es schon seit Uralt-Rauchpilz-TOS-Zeiten so war und auch im STE-TOS 1.6 sowie im TT-TOS 3.01 noch immer so ist. Die Methode funktioniert prächtig, wie der geneigte Leser sicher schon erraten hat, und wird es wohl auch weiterhin, doch kann man dies weder 100%ig noch für alle Zeiten garantieren.

Lösungsvorschlag

Die erste Methode bietet mehr Sicherheit, eignet sich aber nicht für Programme, die, wie das nun vorgestellte *MAL_FIND* oder *GET_MPB*, lediglich eine darstellende Funktion haben. Um nicht umständlich mit Boot-Sektor-Installationen herumbasteln zu müssen, könnte man auch ein Päckchen installieren (ein Bonus für Programme, die sowieso auf einem resetfesten Päckchen laufen sollen!), evtl. gleich einen Warmstart veranstalten und aus dem Päckchen heraus den MPB ermitteln. Da man hier immer damit rechnen muß, daß ein selbst-bootender Harddisk-Treiber von Laufwerk C: dazwischenfunkt, empfiehlt sich eine Kombination der beiden Methoden, wie es in dem als Link-Modul ausgelegten *_GET_MPB* implementiert wurde.

Gesucht wird jeweils von *membot* nach unten bis $2000 (willkürlich, auch bei Uralt- TOS-Versionen findet sich nichts darunter). Hierbei wird vorausgesetzt, daß die Liste komplett im ursprünglichen OS-Pool liegt und dieser nicht etwa schon erweitert wurde (z.B. durch FOLDRXXX.PRG). Ferner ist denkbar, daß Pool-Manipulationen residenter Programme, z.B. Monitore, den Suchalgorithmus blokkieren könnten, die Reihenfolge der Programme wäre hier zu beachten: Je früher der Start, desto höher die Erfolgswahrscheinlichkeit! Die Suchrichtung von oben nach unten ist zwar um einiges langsamer, hat sich aber als stabiler erwiesen. Das Problem der 'Sackgassen durch Geister-MDs' wird zwar in beiden Suchrichtungen von der Routine gleich gut bewältigt, auch wenn man ihnen bei der Suche von unten nach oben natürlich öfter begegnet, zuweilen tauchen jedoch kurzfristig (!) 'verstümmelte' MDs auf, die mit ausgenulltem *m_own* vorzeitig die Abbruchbedingung erfüllen. Diese Gefahr hat sich bei der Suche von oben nach unten nicht ergeben.

Einbruchsgefahr!

Wie man sieht, bewegen wir uns hier auf ziemlich dünnem Eis. Sinnvolle Plausibilitätstests, in der richtigen Kombination, sind demnach das A und O solcher Routinen, da ließe sich evtl. noch einiges verfeinern. Das war auch mit der schwierigste Teil beim Entwurf der Routine, denn hier spiegeln sich ja alle Annahmen, die man so macht. Der Test auf 24-Bit-Adressen mußte dem TT zuliebe z.B. wieder gestrichen werden, ebenso die Annahmen über das Verhalten des *mp_rover*. Andere Tests erwiesen sich als nicht stabil: der 'Adhockery' sind bei solchen Problemen ja Tür und Tor geöffnet, und man muß schon sehr aufpassen, was für Annahmen man macht und was sie implizieren. Man darf sich auch nicht von Speicherabbildern, sog. Dumps, zu generalisierenden Annahmen verleiten lassen. Wenn ich vorhin sagte, daß der Bereich um den MPB so schön leer sei, so gilt das durchaus nicht immer. Beim Nachladen anderer Programme werden u.U. direkt anschließend Tabellen mit Deskriptoren angelegt. Das bedeutet, daß die Abbruchbedingung der Routine z.B. unter auf dem AES laufenden Debuggern oder bestimmten Shells (u.a. Gulam, Gemini 1.2) nicht erfüllt werden kann. So etwas ist aber nun gerade nicht die Umgebung, unter der die Routine nachher laufen soll, und ich habe mich daher gehütet, hier ad hoc Abhilfe zu schaffen!

Weitere Einzelheiten zu erklären, erspare ich mir, und lasse dafür lieber den Code sprechen: Er ist ja kurz genug und, wie ich hoffe, ausreichend kommentiert. Also, immer die Zeiger suchen oder den Zeigern folgen! Glücklicherweise läßt sich dieses ganze Zeigergewurstel in Assembler nicht nur einigermaßen elegant programmieren, sondern der Code ist auch entsprechend schnell in der Ausführung, so daß selbst bei RAM-TOS, wo *membot* ja ziemlich hoch liegt und sich der Algorithmus erst durch das ganze TOS.IMG wühlen muß, passable Ausführungszeiten erreicht werden.

Die Programme

MAL_FIND gibt die ganze *mal* entsprechend dem Algorithmus von hinten nach vorn aus, wobei ungültige MDs mit *<invalid!* gekennzeichnet werden. Weitere solche Markierungen hintereinander bedeuten, daß entsprechend viele Zeilen darüber als ungültig zu betrachten sind, wie schon oben erklärt. Dann wird der MPB ausgegeben und anschließend die *mfl* [8] von vorn nach hinten, den *m_link*-Zeigern folgend. Falls der Suchalgorithmus bei Ausgabe der *mal* steckenbleiben sollte, gibt es eine entsprechende Fehlermeldung.

GET_MPB gibt bei Erfolg nur den MPB aus, sonst eine Fehlermeldung. Der Code entspricht *MAL_FIND* bis auf die MD-Listenausgabe.

_GET_MPB ist ein Modul, das zur Einbindung in Hochsprachen gedacht ist. In D0 wird bei Erfolg die Adresse des MPB zurückgeliefert, sonst eine Null. Man bedenke allerdings, daß so ein Modul wirklich nur zur Vewendung in ganz speziellen (meist resetfesten) Utilities geeignet ist und nicht etwa für Standard TOS- oder gar AES-Applikationen. Dort gibt es schließlich genug bekannte und gut dokumentierte Methoden, sich Speicher zu besorgen.

CART_MPB und *BOOT_MPB* sind die Implementationen der Routine für Cartridge bzw. Boot-Sektor. Sie installieren ein Cookie namens *MPB** im Cookie Jar, auf das andere Programme dann zugreifen können. Falls das Cookie Jar erst eingerichtet werden muß (<TOS 1.6), wird auch ein Resethandler installiert, der die Systemvariable *_p_cookies* $5A0 bei einem allfälligen Warmstart wieder löscht, wie von Atari in [9] vorgeschrieben. Speicher für das Cookie Jar besorgen sich beide Programme in einer Art, die ich 'stille Reservierung' nennen möchte - also ohne Benutzung von *Malloc()*. Das ist, wie man sieht, sehr einfach - wir sitzen ja schließlich an der Quelle! Im Falle der Cartridge ist es noch einfacher und zudem die einzig mögliche Methode, da ja noch keine GEMDOS-Aufrufe gemacht werden dürfen.

Probe aufs Exempel

Spott erntet, wer sich darüber beklagt, daß sein wunderbares XYZ-Programm nicht

auf der neuen OS-Version läuft. Neulich hatte ein (registrierter!) User endlich sein Update eines bekannten biegsamen Programms für TOS 1.4 erhalten und strahlte übers ganze Gesicht. Der STE stand auf dem Tisch. Ich sagte nur: 'Nimm doch mal diesen hier...', 'Unbekannte TOS-Version', das war's! Betretenes Schweigen... -

Bei aller Begeisterung für den wunderbar 'tight & clean' geschriebenen Code des Luftschlosses [10] aus Scheibenkleister II habe ich es wegen der dort enthaltenen Versionsabfragen zweiter Art nicht gustiert. RRN2500F wurde nach kurzem Test wieder gelöscht: Aus den Augen, aus dem Sinn! Doch nun her damit und neu ausgepackt! Das Stückchen Code schnell ersetzt, assembliert und gleich mal auf dem STE gestartet: läuft! Desgleichen auf diversen modernen RAM-TOS-Versionen, Blitter-TOS, KAOS, RAM/ROM-TOS 1.0, Very OLD RAM-TOS 1.0 vom 20.11.85. Läuft nicht auf Rauchpilz-TOS, scheitert dort aber an *act_pd* (nicht in der Liste). Eine Abfrage erster Art wäre hier natürlich möglich, wird jedoch nicht eingearbeitet, da unwichtig!. Wer noch auf so einer alten Gurke hackt, muß wohl irgendwie von der Zivilisation abgeschnitten sein, vielleicht auf Expedition am Amazonas...

Bernd Rosenlecher

P.S. Nach nunmehr über einem Jahr stabilen Dauerbetriebs des mit der hier vorgestellten MPB-Suchroutine ausgerüsteten 'Luftschlosses' auf meinem und einigen anderen Systemen unter wechselnden Konfigurationen, konnte es dank der Nachrüstung des stufenweisen Dereferenzierungsalgorithmus' nun auch erfolgreich auf dem TT030 (32 MHz, TOS 3.01) getestet werden, so daß ich, von der relativen (Beachtung der Caveats) Robustheit des Algorithmus' überzeugt, diesen einer größeren Öffentlichkeit nicht länger vorenthalten möchte. Eigentlich schön, daß Atari zur Zeit so aktiv ist: Die nächste Probe aufs Exempel wäre, ob's auch auf TOS 2.05 des neuen Mega STE läuft.

Anmerkungen und Referenzen:

1. Leider sind diese Methoden selbst bei Leuten, die ansonsten etwas anderes 'predigen', noch immer nicht ganz aus der Mode gekommen. Hat's doch subjektiv auch manches für sich: Das anhängige Problem ist gelöst, und wenn's dann spätestens bei Erscheinen der nächsten OS-Version nicht mehr funktioniert, muß der Programmierer eben das Programm anpassen (Auftrag) und der Kunde halt zahlen (fürs Update). So steigert man das Bruttosozialprodukt!

2. Landon Dyer: A Hitchhiker's Guide to the BIOS (HHG), 1985, Atari Corp

Unter 'themd' heißt es: „Filled in by the BIOS on a 'getmpb' call; indicates to GEMDOS the limits of the TPA. ...(you can't use the m_link field in the first MD). Someday (with a better GEMDOS) these limitations may be lifted.“ ...und stellt somit die Freigabe des MPB in Aussicht.

3. Alex Esser: TOS intern, ST Computer 1987, Sonderheft Nr.2, S.35 ff.:

Umfassende Analyse der GEMDOS-Speicherverwaltung, mit Programmen zur Erforschung der interessierenden Strukturen, basierend auf TOS 1.0. Siehe ebenfalls den 1. Teil des zweiteiligen Beitrags 'Systemvariablen des TOS', ST Computer 11/88, sowie zu TOS 1.4 'Somewhere over the Rainbow', 2 und 3, ST Computer 5-6/90, vom gleichen Autor. In letzterem Beitrag übrigens auch die Beschreibung des oben genannten Pool-Bugs.

4. Allan Pratt, UUCP Message-ID: <2737@atari.UUCP>, 12 Nov 90 20:03:26 GMT „....Getmpb is a BIOS call which is used by GEMDOS to find out the lay of the land at the beginning of the world: it is the BIOS's way of telling GEMDOS what memory it found and where. It should not be used by user programs. ...“

5. Zu diesem Zeitpunkt könnten wir übrigens auch, ohne den MPB zu ermitteln und damit dann die 'mfl' wie gehabt zu bearbeiten, völlig simpel gleich 'themd' manipulieren. Das ist es nämlich, was in Wirklichkeit geschieht, wenn man zu eben der Zeit über den MPB auf die 'mfl' zugreift: Sie besteht dann ja nur aus 'themd'!

6. Allan Pratt, UUCP Message-ID: <2756@atari.UUCP>, 30 Nov 90 02:03:15 GMT „...You can't free memory which has stayed resident as a result of a TSR. It's not even accounted for in the system as „allocated“ memory - it's been UNHOOKED from the memory management lists. It's GONE.“

7. B.Rosenlecher, 'XBRA? XNAM?? BASEFIND!!!', ST Computer 9/90, S.147 ff.

8. Bei meiner Standardkonfiguration (TOS 1.4 mit geladenem POOLFIX3 sowie weiteren fünf residenten und einem nicht residenten AUTO-Ordner-Programm und sechs Accessories) habe ich immer nur einen MD in der 'mfl' gesehen, auf den regelmäßig beide Zeiger mp_mfl und mp_rover zeigten, bei anderen Konfigurationen auch schon mal 2 oder 3, und manchal sogar, daß der mp_rover nicht auf das gleiche Ziel zeigte wie der mp_mfl. Nur einen MD in der 'mfl' zu haben, bedeutet natürlich, daß der als frei ausgewiesene Speicher nur aus einem großen Block besteht und nicht etwa zersplittert ist, eine wirklich erfreuliche Tatsache also!

9. Atari Corp., STE TOS Release Notes, Jan 12, 1990, Sunnyvale, CA 94086

10. Claus Brod, Anton Stepper: Scheibenkleister II, MAXON Computer GmbH, Eschborn 1989. Das 'Luftschloß' ist eine über den Dateinamen konfigurierbare, superschnelle, resetfeste, autobootfähige, winzigkleine (mit eigener Optimierung unter 2 kB) RAM-Disk, die nach Einbau der hier vorgeschlagenen '_get_mpb'-Routine auf allen relevanten TOS-Versionen läuft (hoffentlich zur Freude der Autoren wie auch der Anwender!).


```
 1:  * -------------------------------------------
 2:  * mal_find.s  lists MD's of mal, MPB & MD's of
                               mfl          br 2/90
 3:  * 1st revision: multiple tries in backtracing
                   thru dead MD links       br 1/91
 4:  * 2nd revision: reverse scan direction br 2/91
 5:  * -------------------------------------------
 6:  action:    move.l  4(sp),a0        ;basepage addr
 7:             lea     mystk,a1        ;end of code
 8:             move.l  a1,sp           ;new sp
 9:             suba.l  a0,a1           ;prog length
10:
11:             move.l  a1,-(sp)        ;newsize
12:             move.l  a0,-(sp)        ;block
13:             clr     -(sp)           ;filler
14:             move.w  #$4A,-(sp)      ;Mshrink
15:             trap    #1              ;GEMDOS
16:             lea     $C(sp),sp
17:
18:  start:     dc.w    $A000           ;line_a init
19:             cmpi    #79,-$2C(a0)    ;v_cel_mx:
                             minimum #columns = 80
20:             blt     sorry           ;schade!
```

```
21:             cmpi    #24,-$2A(a0)    ;v_cel_my:
                 minimum #lines = 25
22:             blt     sorry           ;traurig!
23:
24:             lea     title1(pc),a0   ;Titelzeile
25:             bsr     conws           ;ausgeben
26:             lea     subtit(pc),a0   ;Untertitel
27:             bsr     conws           ;ausgeben
28:
29:             pea     main(pc)
30:             move    #$26,-(sp)      ;Supexec
31:             trap    #14             ;XBIOS
32:             addq    #6,sp
33:
34:  term:      bsr     cnecin          ;warte auf Taste
35:             clr     -(sp)           ;Pterm0
36:             trap    #1              ;GEMDOS
37:  * -------------------------------------------
38:  main:      lea     $48E,a3       ;themd = letzter
      MD der mal
39:             lea     $2000,a5      ;Startadresse
40:             movea.l $432,a4       ;Endadresse =
                                       membot        →
```

```
41:          bsr      show_md      ;zum Display
                                    (a3 - > MD)
42:
43:  loop_0:   movea.l a5,a0      ;Startadresse laden
44:            moveq   #0,d6      ;Versuchszähler
45:  loop_1:   addq    #2,a0      ;nur gerade Adressen
46:            cmpa.l  a0,a4      ;Endadresse
             erreicht?
47:            bls     stuck        ;war nichts
48:
49:            cmpa.l  (a0),a3      ;Zeiger da?
50:            bne     loop_1       ;weiter testen
51:
52:  * Plausibilitäts-Tests für MD's, falls (a0) =
     mp_mal - kritisch!
53:
54:            btst    #0,15(a0)    ;m_own gerade? (!)
55:            bne     loop_1       ;weiter testen
56:            btst    #0,7(a0)     ;m_start gerade?
              (!)
57:            bne     loop_1       ;weiter testen
58:            btst    #0,11(a0)    ;m_length gerade?
              (!)
59:            bne     loop_1       ;weiter testen
60:
61:            move.l  a0,a3        ;neuer MD?
62:            bsr     show_md      ;als Eintrag der
                                    'mal' ausgeben
63:
64:  weiter:   tst.l   12(a3)       ;m_own = 0? (!!!)
65:            bne     loop_0       ;nächsten Zeiger
                                    suchen
66:
67:  thats_it: lea     marker(pc),a0 ;als ungültig
68:            bsr     conws        ;markieren
69:            lea     title3(pc),a0 ;neue Überschrift
70:            bsr     conws        ;ausgeben
71:            subq    #4,a3        ;Adresse mpb
72:            move.l  a3,d3        ;umrechnen und
73:            bsr     prt_hex      ;ausgeben
74:            moveq   #5,d4
75:  sp_loop:  bsr.s   space        ;Zwischenraum
76:            dbf     d4,sp_loop
77:            move.l  (a3),d3      ;mp_mfl
78:            bsr     prt_hex      ;ausgeben
79:            bsr.s   space        ;Zwischenraum
80:            move.l  4(a3),d3     ;mp_mal
81:            bsr     prt_hex      ;ausgeben
82:            bsr.s   space        ;Zwischenraum
83:            move.l  8(a3),d3     ;mp_rover
84:            bsr     prt_hex      ;ausgeben
85:            lea     crlf2(pc),a0 ;2 * CR LF
86:            bsr.s   conws        ;ausgeben
87:            lea     title2(pc),a0 ;letzte
                                    Titelzeile
88:            bsr.s   conws        ;ausgeben
89:            lea     subtit(pc),a0 ;Untertitel
90:            bsr.s   conws        ;ausgeben
91:
92:  loop_2:   tst.l   (a3)         ;und jetzt
93:            beq.s   return       ;schön
94:            move.l  (a3),a3      ;einfach
95:            bsr.s   show_md      ;die 'mfl'
                                    ausgeben
96:            bra     loop_2
97:  return:   rts
98:
99:  stuck:    lea     marker(pc),a0 ;als ungültig
100:           bsr.s   conws        ;markieren
101:           addq    #1,d6        ;# Versuche
                                    hochzählen
102:           move.l  a3,a0        ;ab hier weiter
                                    suchen
103:           move.l  (a3),a3      ;wieder alten
                                    Zeiger nehmen
104:           cmp     #10,d6       ;max. Anzahl der
                                    Versuche = 11
105:           bls     loop_1       ;nochmal
                                    versuchen!
106:           lea     sticky(pc),a0 ;leider
107:           bsr.s   conws
                                   ;steckengeblieben!
108:           rts
109: * ----------------------------------------------
110: show_md:  lea     crlf(pc),a0  ;CR LF
111:           bsr.s   conws        ;ausgeben
112:           move.l  a3,d3        ;md_addr
113:           bsr.s   prt_hex      ;ausgeben
114:           bsr.s   space        ;Zwischenraum
            ausgeben
115:           moveq   #3,d5        ;4 Werte
116: next_1:   move.l  (a3)+,d3     ;Wert holen
117:           bsr.s   prt_hex      ;ausgeben
118:           bsr.s   space        ;Zwischenraum
119:           dbf     d5,next_1    ;um durchzuschaun
120:           lea     -16(a3),a3   ;a3 restaurieren
121:           rts
122: * ----------------------------------------------
123: space:    lea     space_1(pc),a0 ;Zwischenraum
124: conws:    pea     (a0)         ;Stringadresse
125:           move    #9,-(sp)     ;Cconws
126:           trap    #1           ;GEMDOS
127:           addq    #6,sp        ;SP restaurieren
128:           rts
129: * ----------------------------------------------
130: cnecin:   move    #7,-(sp)     ;Cnecin
131:           trap    #1           ;GEMDOS
132:           addq    #2,sp
133:           rts
134: * ----------------------------------------------
135: cconout:  move    d0,-(sp)     ;char
136:           move    #2,-(sp)     ;Cconout
137:           trap    #1           ;GEMDOS
138:           addq    #4,sp
139:           rts
140: * ----------------------------------------------
141: * Langwort in d3 in Hex (als Text) auf Konsole
     ausgeben
142:
143: prt_hex:  moveq   #7,d7        ;8 mal
144: nibble:   rol.l   #4,d3        ;jeweils ein
    Nibble
145:           move    d3,d0        ;ans Ende rollen
146:           andi    #$000f,d0    ;isolieren
147:           lea     hextab(pc),a0 ;Hextabelle holen
148:           move.b  0(a0,d0.w),d0 ;und Zeichen
149:           bsr     cconout      ;ausgeben
150:           dbf     d7,nibble    ;weiter
151:           rts
152: * ----------------------------------------------
153: sorry:    lea     lorez(pc),a0 ;'sorry, min
                              screen size 80 * 25!'
154:           bsr     conws        ;ausgeben
155:           bra     term         ;das war's
156: * ----------------------------------------------
157: hextab:   dc.b '0123456789ABCDEF'
158: title1:   dc.b 13,'mal: memory allocated list
          (c) br 90,91',13,10,0
159: subtit:   dc.b    'md_addr    m_link    m_start
      m_length  m_own',13,10
160:           dc.b    '------------------------------
                    --------------------',13,0
161: title2:   dc.b 13,'mfl: memory free list
                 (c) br 90,91',13,10,0
162: title3:   dc.b 13,10,10,'mpb: memory parameter
                 block    (c) br 90,91',13,10
163:           dc.b            'mpb_addr
        mp_mfl    mp_mal    mp_rover', 13,10
164:           dc.b            '------------------------
               ------------------------',13,10,0
165: sticky:   dc.b    13,10,'* * * * * *    sorry,
            got stuck!    * * * * * *',13,10,0
166: space_1:  dc.b    '  ',0
167: crlf2:    dc.b    10
168: crlf:     dc.b    13,10,0
169: marker:   dc.b    '<invalid! ',0
170: lorez:    dc.b    13,10,'sorry, min screen
                         size 80 * 25!',13,10,0
171: * ----------------------------------------------
172:           even
173:           bss
174:           ds.l    100
175: mystk:
```

```
1:  * ----------------------------------------------
2:  * get_mpb.s (the real one!)               br 2/90
3:  * 1st revision: multiple tries in backtracking
      thru dead MD links                     br 1/91
4:  * 2nd revision: reverse scan direction    br 2/91
5:  * ----------------------------------------------
6:  action:    move.l    4(sp),a0       ;basepage addr
7:             lea       stack+$400(pc),a1   ;end of
                                 code + $100 longs
8:             move.l    a1,d0
```

→

```
  9:          andi.b   #$FC,d0       ;long align
 10:          move.l   d0,sp         ;new sp
 11:          suba.l   a0,a1         ;prog length
 12:
 13:          pea      (a1)                ;newsize
 14:          pea      (a0)                ;block
 15:          clr      -(sp)               ;filler
 16:          move     #$4A,-(sp)          ;Mshrink
 17:          trap     #1                  ;GEMDOS
 18:          lea      $C(sp),sp
 19:
 20:          lea      title(pc),a0   ;Titelzeile
 21:          bsr      conws          ;ausgeben
 22:
 23:          pea      main(pc)            ;get_mpb
 24:          move     #$26,-(sp)          ;Supexec
 25:          trap     #14                 ;XBIOS
 26:          addq     #6,sp
 27:
 28:          tst.l    d0                  ;na?
 29:          beq.s    error               ;schade!
 30:
 31:          move.l   d0,a3          ;Adresse mpb
 32:          move.l   a3,d3
 33:          bsr      prt_hex        ;ausgeben
 34:          moveq    #2,d5          ;3 mal
 35: loop_2:  bsr.s    space          ;Zwischenraum
 36:          move.l   (a3)+,d3       ;mp_mfl,
                                 mp_mal, mp_rover
 37:          bsr      prt_hex        ;ausgeben
 38:          dbf      d5,loop_2
 39:
 40:          lea      bye(pc),a0   ;'Taste drücken!'
 41:          bsr.s    conws        ;ausgeben
 42:
 43: term:    move     #7,-(sp)            ;Cnecin
 44:          trap     #1                  ;GEMDOS
 45:          clr      (sp)                ;Pterm0
 46:          trap     #1                  ;GEMDOS
 47:
 48: error:   lea      err_1(pc),a0  ;Fehlermeldung
 49:          bsr.s    conws         ;ausgeben
 50:          bra      term
 51: * ---------------------------------------------
 52: main:    lea      $48E,a3       ;themd = letzter
                                     MD der mal
 53:          lea      $2000,a4      ;Startadresse
 54:          movea.l  $432,a5       ;Endadresse =
                                     membot
 55:
 56: loop_0:  movea.l  a4,a0         ;Startadresse
 57:          moveq    #0,d6         ;Versuchszähler
 58: loop_1:  addq     #2,a0         ;nur gerade
                                     Adressen
 59:          cmpa.l   a0,a5         ;Endadresse
                                     erreicht?
 60:          bls.s    stuck         ;war nichts
 61:
 62:          cmpa.l   (a0),a3       ;Zeiger?
 63:          bne      loop_1        ;weiter testen
 64:
 65:          btst     #0,15(a0)     ;m_own gerade?
 66:          bne      loop_1        ;weiter testen
 67:          btst     #0,7(a0)      ;m_start gerade?
 68:          bne      loop_1        ;weiter testen
 69:          btst     #0,11(a0)     ;m_length gerade?
 70:          bne      loop_1        ;weiter testen
 71:
 72:          move.l   a0,a3         ;neuer MD?
 73:          tst.l    12(a3)        ;m_own = 0
           Besitzer? (!!!)
 74:          bne      loop_0        ;nächster Zeiger
 75:
 76:          subq     #4,a3         ;Adresse des MPB
 77:          move.l   a3,d0         ;Rückgabewert
 78:          rts
 79:
 80: stuck:   addq     #1,d6         ;# Versuche
                                     hochzählen
 81:          move.l   a3,a0         ;ab hier
                                     weitersuchen
 82:          move.l   (a3),a3       ;wieder alten
                                     Zeiger nehmen
 83:          cmp      #10,d6        ;max. Anzahl der
                                     Versuche = 11
 84:          bls      loop_1        ;nochmal
                                     versuchen!
 85:          moveq    #0,d0         ;Fehler
 86:          rts
 87: * ---------------------------------------------
 88: space:   lea      space_1(pc),a0 ;Zwischenraum
 89: conws:   pea      (a0)          ;Stringadresse
 90:          move     #9,-(sp)      ;Cconws
 91:          trap     #1            ;GEMDOS
 92:          addq     #6,sp        ;SP restaurieren
 93:          rts
 94: * ---------------------------------------------
 95: cconout: move     d0,-(sp)            ;char
 96:          move     #2,-(sp)            ;Cconout
 97:          trap     #1                  ;GEMDOS
 98:          addq     #4,sp
 99:          rts
100: * ---------------------------------------------
101: * Langwort in d3 in Hex (als Text) auf Konsole
     ausgeben
102:
103: prt_hex: moveq    #7,d7      ;8 mal jeweils
104: nibble:  rol.l    #4,d3      ;ein Nibble
105:          move     d3,d0      ;ans Ende rollen
106:          andi     #$000f,d0  ;isolieren
107:          lea      hextab(pc),a0
                                 ;Hextabelle holen
108:          move.b   0(a0,d0.w),d0  ;und Zeichen
109:          bsr      cconout        ;ausgeben
110:          dbf      d7,nibble      ;weiter
111:          rts
112: * ---------------------------------------------
113: hextab:  dc.b '0123456789ABCDEF'
114: title:   dc.b   13,'get_mpb (the real one!)
                 (c) br 90,91',13,10,10
115:          dc.b     'mpb_addr  mp_mfl    mp_mal
             mp_rover',13,10
116:          dc.b     '---------------------------
                  -----------',13,10,0
117: space_1: dc.b     ' ',0
118: bye:     dc.b  13,10,10,"* * *  press any
     key!       * * *",13,10,0
119: err_1:   dc.b          "* * *   sorry, could't
                     get mpb!  * * *",13,10,0
120:          even
121: stack:
122: * ---------------------------------------------
123:          bss
124:          ds.l     100
```

```
  1: * ---------------------------------------------
  2: * _get_mpb.s link module dev'd from get_mpb (the
     real one!)                               br 2/90
  3: * ---------------------------------------------
  4: * 1st revision: multiple tries in backtracking
                   thru dead MD links       br 1/91
  5: * 2nd revision: split mfl/mal search, reverse
                   first scan direction     br 2/91
  6: * ---------------------------------------------
  7: * long _get_mpb(void);  IN: nothing, OUT:
     address of MPB or NULL in D0.L
  8: * ---------------------------------------------
  9: themd      = $48E
 10: membot     = $432
 11: memtop     = $436
 12: MAX        = 10      ;max # Versuche = 11
 13: * ---------------------------------------------
 14:            globl      _get_mpb            ;C &
             assembly language entry
 15:            globl      GETMPB         ;FORTRAN
             & Pascal entry
 16: GETMPB:
 17: * ---------------------------------------------
 18: _get_mpb: movem.l   d1-d2/a0-a2,-(sp)
                                 ;Register retten
 19:           pea       get_mpb(pc)         ;Routine
 20:           move      #$26,-(sp)          ;Supexec
 21:           trap      #14                 ;XBIOS
 22:           addq      #6,sp
 23:           movem.l   (sp)+,d1-d2/a0-a2
                                 ;Register zurück
 24:           rts
 25: * ---------------------------------------------
 26: get_mpb:  lea     themd,a1    ;Ausgangspunkt
 27:           movea.l membot,a2   ;Suchende
 28:           move.l  memtop,d0
 29:           sub.l   a2,d0       ;freien Speicher
```

→

```
                                 berechnen
30:          cmp.l    8(a1),d0   ;m_length = memtop
                                           - membot?
31:          bne.s    loop_0     ;fehlt schon etwas
32:
33:          lea      $4000,a0   ;Suchbeginn
34: t_loop:  addq     #2,a0      ;nur gerade Adressen
35:          cmpa.l   a0,a2      ;Endadresse erreicht?
36:          bls.s    error      ;fertig
37:
38:          cmpa.l   (a0),a1    ;Zeiger auf themd?
39:          bne      t_loop     ;weiter suchen
40:          bra.s    fini       ;das war's
41:
42: loop_0:  move.l   a2,a0      ;Suchbeginn = membot
43:          moveq    #0,d2      ;Versuchszähler
44: loop_1:  subq     #2,a0      ;nur gerade Adressen
45:          cmpa.l   #$2000,a0  ;Endadresse erreicht?
46:          bls.s    stuck      ;war nichts
47:
48:          cmpa.l   (a0),a1    ;Zeiger da?
49:          bne      loop_1     ;weiter testen
50:
51: * Plausibilitäts-Tests für MD's, falls mp_mal
    gefunden wurde, kritisch!
52:
53:          btst     #0,15(a0)  ;m_own gerade?
54:          bne      loop_1     ;weiter testen
55:          btst     #0,7(a0)   ;m_start gerade?
56:          bne      loop_1     ;weiter testen
57:          btst     #0,11(a0)  ;m_length gerade?
58:          bne      loop_1     ;weiter testen
59:
60:          move.l   a0,a1      ;evtl. neuer MD
61:          tst.l    12(a0)     ;Besitzer = 0?
62:          bne      loop_0
63:
64:          subq     #4,a0      ;Adresse des MPB
65: fini:    move.l   a0,d0      ;Rückgabewert
66:          rts
67:
68: stuck:   addq     #1,d2      ;# Versuche
                                 hochzählen
69:          move.l   a1,a0      ;war kein gültiger
                                 m_link
70:          move.l   (a1),a1    ;alten Zeiger
                                 nehmen
71:          cmp      #MAX,d6    ;max. Anzahl der
                                 Versuche
72:          bne      loop_1     ;nochmal versuchen!
73: error:   moveq    #0,d0      ;Fehler
74:          rts
```

```
 1: * ----------------------------------------------
 2: * cart_mpb.s install 'MPB*' cookie from type #1
    cartridge                                br 1/91
 3: * ----------------------------------------------
 4: p_cookie  = $5A0
 5: longfram  = $59E
 6: resvalid  = $426
 7: resvecto  = $42A
 8: RESMAGIC  = $31415926
 9: membot    = $432
10: trap13    = $B4
11: COOKIE    = 'MPB*'
12: * ----------------------------------------------
13: ca_magic: dc.l     $ABCDEF42
14: ca_next:  dc.l     0
15: ca_init:  dc.l     $02FA0026
16: ca_run:   dc.l     $00FA0026
17: ca_time:  dc.w     0
18: ca_date:  dc.w     0
19: ca_size:  dc.l     ende-start
20: ca_name:  dc.b     'CART_MPB.003',0,0
21: * ----------------------------------------------
22: start:    move.l   membot,a0    ;erstmal Platz
                                    besorgen
23:           move.l   trap13,d0
24:           move.l   d0,(a0)+     ;alten BIOS-Vektor
                                    retten
25:           move.l   a0,membot
26:           lea get_mpb(pc),a0    ;neuen BIOS-Vektor
27:           move.l   a0,trap13    ;installieren
28:           rts
29: * ----------------------------------------------
```

```
 30: get_mpb:   move.l   sp,a0         ;OS ist immer im
     Super
 31:            tst      longfram      ;Prozessor?
 32:            beq.s    weiter
 33:            addq     #2,a0
 34: weiter:    tst      6(a0)         ;Getmpb?
 35:            beq.s    do_it
 36:            move.l   membot,a0     ;dort ist
 37:            subq     #4,a0         ;alter Vektor
 38:            jmp      (a0)
 39: * ---------------------------------------------
 40: * Here's where the action is: IN: p_mpb @ 8(a0),
     OUT: entry in cookie jar
 41: * ---------------------------------------------
 42: do_it:     move.l   8(a0),a0      ;p_mpb
 43:            movem.l  d3-d7/a3-a7,-(sp) ; retten
 44:            move.l   membot,a6     ;altes membot
                                       retten
 45:            move.l   a0,a5         ;MPB merken
 46:
 47:            move.l   #COOKIE,d3    ;'MPB*'
 48:            movea.l  p_cookie,a0   ;*p_cookie
 49:            move.l   a0,d0         ;NULL = TOS <1.6?
 50:            beq.s    make_jar      ;cookie jar nicht
                                       vorhanden
 51:
 52:            moveq    #0,d1         ;Zähler
 53: jarscan:   addq     #1,d1         ;Eintrag zählen
 54:            cmp.l    (a0),d3       ;cookie schon da?
 55:            move.l   (a0)+,d0      ;cookie
 56:            beq.s    put_cook      ;eintragen
 57:            addq     #4,a0         ;nächstes cookie
 58:            bra      jarscan       ;prüfen
 59:
 60: put_cook:  cmp.l    (a0),d1       ;noch Platz?
 61:            beq.s    mak_spc       ;nein
 62:            move.l   (a0)+,d0      ;# d.Einträge
 63:            clr.l    (a0)+       ;Endeintrag
 64:            move.l   d0,(a0)     ;verschieben
 65:            subq     #8,a0       ;zurückpositionieren
 66:            move.l   a5,(a0)     ;Adresse des MPB
        eintragen
 67:            move.l   d3,-(a0) ;cookie eintragen
 68:            bra.s    return   ;unlink from trap etc.
 69:
 70: mak_spc:   movea.l  p_cookie,a1 ;*p_cookie
 71:            move.l   a6,a0         ;Adresse = membot
 72:            move.l   d1,d2         ;Zähler
 73:            add.l    d2,d2         ;für je 2 longs
 74:            subq.l   #3,d2         ;präparieren
 75: j_loop:    move.l   (a1)+,(a0)+   ;altes cookie jar
 76:            dbf      d2,j_loop     ;umkopieren
 77:            move.l   d1,d0        ;minus Endeintrag
 78:            addq.l   #7,d0        ;und um 8 erweitern
 79:            bra.s    _injar
 80:
 81: make_jar:  move.l   a6,a0         ;Adresse = membot
 82:            moveq    #8,d0         ;# Einträge
 83: _injar:    move.l   d3,(a0)+      ;cookie
 84:            move.l   a5,(a0)+      ;Adresse des MPB
 85:            clr.l    (a0)+         ;Endeintrag
 86:            move.l   d0,(a0)+      ;Anzahl eintragen
 87:            move.l   a6,p_cookie   ;cookie jar
                                       eintragen
 88:            lea      48(a0),a0     ;+6*8 position.
 89:
 90: _unjar:    move.l   resvecto,(a0)+ ;Reihenfolge
 91:            move.l   resvalid,(a0)+ ;und Lage wie
                                        unten!
 92:            movea.l  a0,a2            ;Position merken
 93:            lea      reshand(pc),a1 ;Resethandler
 94:            moveq    #copy_cnt,d0
 95: c_loop:    move.l   (a1)+,(a0)+     ;kopieren
 96:            dbf      d0,c_loop
 97:
 98:            move.l   a0,membot     ;neues membot
 99:            move.l   a2,resvecto
100:            move.l   #RESMAGIC,resvalid
101:
102: return:    lea      -4(a6),a0     ;alten trap13
     Vektor
103:            move.l   (a0),trap13 ;restaurieren
104:            movem.l  (sp)+,d3-d7/a3-a7 ;restaur.
105:            jmp      (a0)
106: * ---------------------------------------------
107: vecsave:   dc.l     0             ;Reihenfolge
108: valsave:   dc.l     0             ;und Lage beachten! →
```

```
109:  reshand:  clr.l   p_cookie     ;für < TOS 1.6
110:            move.l  valsave(pc),resvalid
111:            move.l  vecsave(pc),resvecto
112:            jmp     (a6)
113:  copy_cnt  = (* - reshand)/4
114:  * ---------------------------------------------
115:  ende:
```

```
  1:  * ---------------------------------------------
  2:  * boot_mpb.s install 'pMPB' cookie from
      bootsector of drive A:                  br 1/91
  3:  * ---------------------------------------------
  4:  * Bedingungen: Laufwerk A:, Speicher in
                  Originalkonfiguration, sonst raus.
  5:  * Annahme: Es gebe zur Zeit der Ausführung des
              Bootsektors A: einen Zeiger
  6:  * auf 'themd' zwischen SRCHBG und membot
      (Suchrichtung!), den 'MPB.mp_mfl'.
  7:  * ---------------------------------------------
  8:  p_cookie  = $5A0
  9:  resvalid  = $426
 10:  resvecto  = $42A
 11:  RESMAGIC  = $31415926
 12:  themd     = $48E
 13:  membot    = $432
 14:  memtop    = $436
 15:  bootdev   = $446
 16:  COOKIE    = 'MPB*'
 17:  SRCHBG    = $4000
 18:  * ---------------------------------------------
 19:            bra.s   start
 20:            dcb.w   16,'**'
 21:  msg_1:    dc.b    ' MPB 0x',0
 22:  msg_2:    dc.b    ' cookie in jar br91',13,10,0
 23:            dcb.w   16,'**'
 24:  * ---------------------------------------------
 25:  start:    move    #$19,-(sp)   ;Dgetdrv
 26:            trap    #1           ;GEMDOS
 27:            addq    #2,sp
 28:            tst     d0           ;Laufwerk A:?
 29:            bne.s   getout       ;nein
 30:
 31:  get_mpb:  movea.l membot,a6    ;merken
 32:            move.l  memtop,d0    ;freien Speicher
 33:            sub.l   a6,d0        ;berechnen
 34:            lea     themd,a3     ;themd sei
                                   einziger MD der mfl
 35:            cmp.l   8(a3),d0     ;m_length = memtop
                                      - membot?
 36:            bne.s   getout       ;Speicher schon
                              manipuliert, von wem???
 37:
 38:
 39:            lea     SRCHBG,a0    ;willkürliche
                                Startadresse für Suche
 40:            movea.l a6,a1  ;Endadresse = membot
 41:  t_loop:   addq    #2,a0  ;nur gerade Adressen
 42:            cmpa.l  a0,a1  ;Endadresse erreicht?
 43:            bls.s   getout ;fertig
 44:
 45:            cmpa.l  (a0),a3      ;Zeiger da?
 46:            bne     t_loop       ;weiter suchen
 47:
 48:            movea.l a0,a5        ;Adresse des MPB
                                      merken
 49:
 50:            move.l  #COOKIE,d7   ;'MPB*'
 51:            movea.l p_cookie,a0  ;*p_cookie
 52:            move.l  a0,d0        ;NULL = TOS <1.6?
 53:            beq.s   make_jar     ;cookie jar nicht
                                      vorhanden
 54:
 55:            moveq   #0,d1        ;Zähler
 56:  jarscan:  addq    #1,d1        ;Eintrag zählen
 57:            cmp.l   (a0),d7      ;cookie schon da?
 58:            beq     tell_it      ;das war's schon
 59:            move.l  (a0)+,d0     ;cookie
 60:            beq.s   put_cook     ;hier eintragen
 61:            addq    #4,a0        ;nächstes cookie
 62:            bra     jarscan      ;prüfen
 63:
 64:  put_cook: cmp.l   (a0),d1      ;noch Platz?
 65:            beq.s   mak_spc      ;nein
 66:            move.l  (a0)+,d0     ;Zahl der Einträge
 67:            clr.l   (a0)+        ;Endeintrag
 68:            move.l  d0,(a0)      ;verschieben
 69:            subq    #8,a0      ;zurückpositionieren
 70:            move.l  a5,(a0)     ;Adresse des MPB
                                         seintragen
 71:            move.l  d7,-(a0)        ;cookie eintragen
 72:            bsr.s   tell_it         ;Meldung ausgeben
 73:  getout:   rts                     ;fertig
 74:
 75:  mak_spc:  movea.l p_cookie,a1 ;*p_cookie
 76:            move.l  a6,a0       ;Adresse = membot
 77:            move.l  d1,d2       ;Zähler
 78:            add.l   d2,d2       ;für je 2 longs
 79:            subq.l  #3,d2       ;präparieren
 80:  j_loop:   move.l  (a1)+,(a0)+ ;altes cookie jar
 81:            dbf     d2,j_loop   ;umkopieren
 82:            move.l  d1,d0      ;minus Endeintrag
 83:            addq.l  #7,d0      ;und um 8 erweitern
 84:            bra.s   _injar
 85:
 86:  make_jar: move.l  a6,a0       ;Adresse = membot
 87:            moveq   #8,d0       ;# Einträge
 88:  _injar:   move.l  d7,(a0)+    ;cookie
 89:            move.l  a5,(a0)+    ;Adresse des MPB
 90:            clr.l   (a0)+       ;Endeintrag
 91:            move.l  d0,(a0)+    ;Anzahl eintragen
 92:            move.l  a6,p_cookie ;cookie jar
                                      eintragen
 93:            lea     48(a0),a0  ;+6*8 positionieren
 94:
 95:  _unjar:   move.l  resvecto,(a0)+ ;Reihenfolge
 96:            move.l  resvalid,(a0)+ ;und Lage wie
                                        unten!
 97:            movea.l a0,a2          ;Position merken
 98:            lea     reshand(pc),a1 ;Resethandler
 99:            moveq   #copy_cnt,d0
100:  c_loop:   move.l  (a1)+,(a0)+    ;kopieren
101:            dbf     d0,c_loop
102:
103:            move.l  a0,membot     ;neues membot
104:            addq    #4,a3         ;m_start
105:            move.l  a0,(a3)+      ;eintragen
106:            suba.l  a6,a0         ;Platz
107:            move.l  a0,d0         ;besorgen
108:            sub.l   d0,(a3)       ;neues m_length
109:
110:            move.l  a2,resvecto
111:            move.l  #RESMAGIC,resvalid
112:
113:  tell_it:  lea     msg_1(pc),a0 ;string
114:            bsr.s   message
115:            move.l  a5,d3          ;MPB
116:            bsr.s   prt_hex
117:            lea     msg_2(pc),a0 ;string
118:
119:  message:  pea     (a0)          ;string
120:            move    #9,-(sp)      ;Cconws
121:            trap    #1            ;GEMDOS
122:            addq    #6,sp
123:            rts
124:  * ---------------------------------------------
125:  cconout:  move    d0,-(sp)           ;char
126:            move    #2,-(sp)           ;Cconout
127:            trap    #1                 ;GEMDOS
128:            addq    #4,sp
129:            rts
130:  * ---------------------------------------------
131:  * Wort in d3 in Hex (als Text) auf Konsole
      ausgeben
132:
133:  prt_hex:  moveq   #3,d7      ;4 mal
134:  nibble:   rol     #4,d3      ;jeweils ein Nibble
135:            move    d3,d0      ;ans Ende rollen
136:            andi    #$000f,d0  ;isolieren
137:            lea     hextab(pc),a0 ;Hextabelle
                                       holen
138:            move.b  0(a0,d0.w),d0 ;und Zeichen
139:            bsr     cconout        ;ausgeben
140:            dbf     d7,nibble      ;weiter
141:            rts
142:  * ---------------------------------------------
143:  vecsave:  dc.l    0         ;Reihenfolge
144:  valsave:  dc.l    0         ;und Lage beachten!
145:  reshand:  clr.l   p_cookie  ;für < TOS 1.6
146:            clr     bootdev   ;von Floppy booten
      bei < TOS 1.6
147:            move.l  valsave(pc),resvalid
148:            move.l  vecsave(pc),resvecto
149:            jmp     (a6)
150:  copy_cnt  = (* - reshand)/4   ;1 added in loop
151:  * ---------------------------------------------
152:  hextab:   dc.b    '0123456789ABCDEF'
```



DFÜ und der Rest der Welt

Modem oder Akustikkoppler

Der Ausgangspunkt einer spannenden und informativen Reise in die Welt der Datenfernübertragung ist die 25polige serielle Schnittstelle an der Rückseite des Atari-Computers. Nun fehlt nur noch ein Akustikkoppler oder ein Modem, das an diese Schnittstelle angeschlossen wird, um die Daten für eine Übertragung auf dem Telefonnetz aufzubereiten. Diese Geräte setzen die digitalen Daten des Computers in analoge Signale um und umgekehrt. In dieser Folge unserer DFÜ-Ecke wollen wir Ihnen zeigen, welche Vor- und Nachteile die beiden Systeme haben und worauf Sie beim Einkauf eines Modems oder Akustikkopplers achten müssen.

Bis vor ein paar Jahren lief die Datenfernübertragung fast ausschließlich mit Hilfe der Akustikkoppler, da die Modems fast unerschwinglich waren. Besonders das Monopol der Post bei den Telekommunikationsendgeräten blockierte den Vormarsch der Modems. Ein Blick in die USA oder ins europäische Ausland beweist das. Da die Akustikkoppler nicht direkt an das Telefonnetz angeschlossen werden, fielen die Koppler in der Bundesrepublik nicht unter das Postmonopol. Mit dem 1. Juli 1990 trat dann die Wende ein: Die Post verzichtete auf das einträgliche Endgerätemonopol, nicht nur bei den Telefonapparaten. Seit dieser Zeit kann jeder Computer-Anwender mit einem Modem seinen Computer mit dem Telefonnetz verbinden. Allerdings ist die Auswahl bei den Modems noch eingeschränkt. Die Post verbietet, daß Modems angeschlossen werden, die nicht für das Netz der Telekom zugelassen sind und keine ZZF-Prüfnummer haben.

Trotz des erheblichen Preisverfalls bei den Modems sind die Akustikkoppler in der Anschaffung in der Regel deutlich billiger. Dieser Preisvorteil kann aber im Laufe der Zeit durch die höheren Telefongebühren dahinschmelzen, denn mit Akustikkopplern sind nicht allzu schnelle Übertragungsgeschwindigkeiten möglich. Übertragungsraten von 300 bis 1200 Baud sind üblich, einige wenige Akustikkoppler schaffen auch 2400 Baud. Bei diesen hohen Geschwindigkeiten ist die Übertragung mit dem Akustikkoppler allerdings recht fehleranfällig. Doch die Arbeit mit einem Akustikkoppler hat auch Vorteile: Man braucht keinen besonderen Anschluß für das Telefonnetz, etwa eine eigene TAE-Telefonbuchse, die bei der Post zusätzliche Gebühren kostet. Akustikkoppler sind auch sehr einfach beim mobilen Einsatz zu gebrauchen. Die meisten Koppler kommen mit der Spannung aus, die der Rechner liefert, oder sind mit Batterien für den Einsatz weit weg von einer Steckdose ausgerüstet.

Bei den Akustikkopplern erfolgt - wie der Name es andeutet - Anbindung an das Telefonnetz akustisch; die Signale werden über die Luft als hörbare Töne übertragen. Dadurch ist diese Übertragungsart auch durch Umgebungsgeräusche beeinflußt, so daß man sich auf eine andere Übertragungsart besann. In manchen Akustikkopplern kann neben der akustischen auch eine induktive Kopplung genutzt werden, es werden also nicht die Töne, sondern die magnetischen Felder, die beim Erzeugen der Töne entstehen, ausgewertet. So werden die Störgeräusche, etwa ein Radio im Hintergrund, ausgeschaltet. Bei Kopplern

dieser Art kann die Art der Übertragung (akustisch/induktiv) eingestellt werden, da nicht jedes Telefon mit magnetischen Hörkapseln ausgerüstet ist.

Mit Akustikkopplern ist Datenfernübertragung nicht gerade komfortabel: Bevor man die Verbindung zu einem anderen Rechner herstellen kann, muß zunächst der Hörer des Telefons in die Gummimuffen des Kopplers gepreßt werden. Probleme tauchen auf, wenn beispielsweise der Telefonhörer eines modernen Designer-Telefons nicht einwandfrei in diese Muffen paßt. Weiterer Nachteil: Bei den Kopplern muß man die Nummer der Mailbox am Telefon per Hand wählen.

Das Modem bietet eine wesentlich bessere Übertragungsqualität als der Akustikkoppler. Es wird mit einem Kabel mit der Telefonbuchse verbunden und damit direkt an das Telefonnetz angekoppelt. Beim Modembetrieb braucht man nicht mehr (wie beim Akustikkoppler) selbst zu wählen. Nicht nur vergeßliche DFÜler wissen das zu schätzen. Das Modem ist also vom Computer aus mit Befehlen ansteuerbar, und es kann auch selbst die Leitung bei einem Anruf abheben. Durch die größere Datensicherheit sind höhere Übertragungsraten bis hin zu 14.400 Bit/s sowie zusätzliche Sicherungsmechanismen möglich.

Auf dem Computer-Markt, insbesondere in den USA, haben sich die Geräte des amerikanischen Herstellers Hayes durchgesetzt. Inzwischen richten sich auch andere Hersteller nach dem Befehlssatz der Hayes-Geräte, dem sogenannten Hayes-Standard. Dieser Standard ist in den fundamentalen Befehlen (Wählen, Abheben, Auflegen, Reset...) bei jedem Hayes-kompatiblen Modem gleich. Durch die verschiedenen Erweiterungen, die jeder Modemhersteller bei seinem Modem eingebaut hat, gibt es aber auch Variationen, die im Einzelfall zu Problemen führen können.

Die wichtigsten Kommandos der Hayes-kompatiblen Modems

Hayes-Befehle beginnen, von zwei Ausnahmen abgesehen, immer mit den Zeichen AT (von attention= engl.: Achtung)

ATZ	Modem-Reset (Modem antwortet mit OK)
ATD	Wahlbefehl
ATDP	Wählen mit dem (bei uns üblichen) Impulswahlverfahren Beispiel: atdp066567874 (wählt die Zerberus-Box FULMIN an). Muß etwa bei einer Nebenstellenanlage erst eine Null vorweg gewählt werden, kann man den Hayes-Befehl W (Warten auf das Freizeichen) einsetzen: ATDP 0 W 066567874. Man kann auch mit dem Befehl ',' (Pause während der Wahl einlegen) arbeiten: ATDP 0 ,, 066567874.
ATDT	Wählen mit dem (besonders in den USA verbreiteten) Tonfrequenzverfahren
ATDS	Eine im Modem gespeicherte Telefonnummer wird angewählt.
ATSO=1	Das Modem antwortet nach einem Klingelzeichen.

Die beiden Nicht-AT-Befehle:

A/	Der zuletzt eingegebene Befehl wird wiederholt.
+++	Bricht Online-Modus ab, das Modem kehrt in den Kommandomodus zurück. Allerdings darf vor und nach der Eingabe der drei Plus-Zeichen eine Sekunde lang kein Datentransfer stattfinden. Das soll verhindern, daß das Modem abbricht, wenn per Zufall drei Plus-Zeichen in einem zu übertragenden Text auftauchen.

CCITT-Normen

Die verschiedenen Übertragungsarten, die ein Modem oder Akustikkoppler beherrschen sollte, sind durch das CCITT genormt. Das Consultative Committee on International Telegraphy and Telephony (CCITT) ist die Unterorganisation der Vereinten Nation für Fernmeldefragen, in der Fernmeldeunternehmen aus aller Welt vertreten sind. Das CCITT stellt also eine Art DIN für alles dar, was mit Datenübertragung und Kommunikation zu tun hat. Dort werden beispielsweise in den V-Normen Übertragungsarten für Modems festgelegt. Die V-Normen, mit denen die Modems ausgezeichnet werden, zeigen, welche Geschwindigkeiten und Betriebsarten vondem Modem beherrscht werden. Die wichtigsten sind:

V.17:	14400/12000 bit/s halbduplex für Gruppe-3 Fax
V.21:	300 bit/s duplex
V.22:	1200 bit/s duplex
V.22bis:	2400 bit/s duplex
V.23:	1200/75 bit/s bzw. 75/1200 bit/s halbduplex
V.29:	9600/7200/4800 bit/s halbduplex für Gruppe-3 Fax
V.32:	9600/4800 bit/s duplex
V.32bis:	14400/12000/9600/7200/4800 bit/s duplex

Bei mehreren Geschwindigkeitsangaben, etwa bei der Norm V 17, besteht die Möglichkeit eines Fallbacks: Das Modem kann hier bei einer schlechten Telefonleitung etwa von 14400 bit/s auf 12000 bit/s zurückschalten, um dann weniger empfindlich gegen Störungen zu sein. Diese Fallback-Möglichkeit haben viele Modems. Ein Problem ist allerdings, daß die wenigsten Modems in der Lage sind, nach einer bestimmten Zeit (wenn die Leitung wieder besser ist) auch wieder zurück auf die höhere Geschwindigkeit zu gehen.

Btx-Nutzer aufgepaßt

DFÜ-Einsteiger, die mit dem Modem auch mit dem Bildschirmtext (Btx) der Post arbeiten wollen und nicht in einer größeren Stadt wohnen, müssen besonders darauf achten, daß ein Modem auch die V.23-Norm (1200/75 bit/s) beherrscht. Dieser Btx-Zugang ist in allen Ortsnetzen unter der Nummer 190 (bzw. 01910 in ländlichen Gebieten) zu erreichen.

In den Ortsnetzen von Augsburg, Bayreuth, Bielefeld, Bonn, Bremen, Detmold, Dortmund, Essen, Freiburg, Gießen, Kaiserslautern, Karlsruhe, Kassel, Kiel, Koblenz, Köln, Krefeld, Mainz, Mannheim, Meschede, München, Norden, Offenburg, Osnabrück, Recklinghausen, Regensburg, Saarbrücken, Salzgitter, Singen und Würzburg bietet die Post unter der Nummer 19300 einen Btx-Zugang mit 1200/1200 bit/s und unter 19304 einen Zugang mit 2400/2400 bit/s an. In Münster/Wesf. und Wiesbaden kann man bislang nur mit 1200 Baud den Btx-Rechner anwählen. In den Großstädten Berlin, Düsseldorf, Frankfurt/Main, Hamburg, Hannover, München, Nürnberg und Stuttgart gibt es einen turboschnellen ISDN-Zugang mit 9600/9600 bit/s (Telefonnummer 19306).

Btx-Nutzer, die in ihrem Ortsnetz keinen 1200- oder 2400-Baud-Zugang haben, können sich natürlich per Ferngespräch in die Ortsnetze einwählen, die diese schnellen Btx-Zugänge bieten. Auf die Dauer treibt aber dieser Btx-Betrieb die Telefonrechnung in ungeahnte Höhen. Ein Modem, das den V.23-Modus beherrscht, ist für diese Benutzergruppe ohne direkten 1200/2400-Baud-Zugang zum Btx sicherlich die bessere Lösung. Die Post hat versprochen, bis Ende 1992 einen bundesweit einheitlichen Zugang mit einer Telefonnummer für alle Baud-Raten anzubieten.

High-Speed-Modems

Hochgeschwindigkeitsmodems nutzen meist eigene Standards, die nicht vom CCITT genormt sind, und oft nur von den jeweiligen Herstellern selbst genutzt werden. Die Post bietet beispielsweise ein Modem mit 19200 Baud an, das nur bis zu 2400 bit/s der CCITT-Norm entspricht. Eine Verbindung mit höheren Baud-Raten ist nur zwischen den baugleichen Modems möglich oder mit den Geräten der Original-Herstellerfirma Telebit. Mittlerweile setzen sich aber die CCITT-Normen auch in den höheren Übertragungsgeschwindigkeiten durch. Die ersten V.32bis-Modems sind schon kurz nach der offiziellen Verabschiedung der Norm im Frühjahr dieses Jahres auf den Markt gekommen, und sogar die Post will Mitte des Jahres endlich ein Modem mit V.32 anbieten, hat aber bis jetzt noch kein Modem mit V.32bis zugelassen.

Aber nicht nur die Übertragungsraten im Modembereich sind in den V-Normen festgelegt, sondern auch alles andere, was mit nicht-paketvermittelnder Übertragung zu tun hat:

- V.25: Modem mit gesonderter Steckverbindung zum Wählen
- V.25bis: Modem mit automatischer Wähleinrichtung über serielle Schnittstelle - man kann mit Kommandos wählen.
- V.24: Schnittstelle zwischen Modem und Rechner - sie umfaßt etwa 55 verschiedene Leitungen, die zur sinnvollen Steuerung zwischen zwei Partnern bestehen können. Es werden nicht nur Sende-, sondern auch Steuer-, Melde- und Taktleitungen definiert, alle verwenden aber einen gemeinsamen Rückleiter.
- V.28: elektrische Schnittstelle zur Datenübertragung - Länge maximal 15 Meter - Spannung zwischen -15V und 15V - „0" bedeutet positive Spannung zwischen 5 und 15V beim Senden und zwischen 3 und 15V beim Empfangen - „1" bei negativer Spannung - Übertragungsrate bis zu 20.000 bit/s
- V.42: Fehlerkorrekturverfahren zwischen zwei Modems, gleichzeitig werden die Daten synchron zwischen den Modems übertragen, können aber asynchron zum Modem kommen.
- V.42bis: Datenkompressionsverfahren. Man erreicht einen bis zu vierfach höheren Durchsatz bei entsprechenden ungepackten Daten.

MNP und V-Normen

Die gerade erwähnte Norm V.42 ist kompatibel zum MNP-Protokoll der amerikanischen Firma Microcom. Diese Firma hat mit ihrem Microcom Networking Protocol (MNP) auf dem Markt den Standard für die Fehlersicherung Hayes-kompatibler Modems gesetzt. Durch ständige Weiterentwicklung gibt es inzwischen verschiedene Stufen des Protokolls, die allgemein als MNP-Klassen bezeichnet werden. Die MNP-Klassen sind untereinander kompatibel, das heißt, ein Modem der MNP Klasse 3 kann ohne Probleme mit einem MNP-5-Modem kommunizieren.

Im MNP-Modus werden die Daten bitorientiert und synchron übertragen. Wie das funktioniert, haben wir in der Juni-Ausgabe der ST-Computer erläutert. Zur Fehlerkorrektur werden die Daten dann zu Paketen zusammengefaßt und durch eine 16 Bit große Prüfzahl gesichert. Durch dieses Verfahren werden also nicht nur Übertragungsfehler minimiert, sondern wird auch die Übertragungsgeschwindigkeit erhöht. Bei der Klasse 3 wird eine effektive Datengeschwindigkeit von 108 Prozent gegenüber einer Übertragung ohne MNP erreicht. In der MNP-Klasse 4, bei der entsprechend der Qualität der Telefonleitung die Länge der MNP-Pakete angepaßt werden, verbessert sich die Effektivität auf 120 Prozent. MNP-Modems der Klasse 5 schicken die Daten durch eine komplizierte Kompressionsroutine und senden sie blockweise. Dadurch kann sich die effektive Datengeschwindigkeit auf das Doppelte der eigentlichen Übertragungsgeschwindigkeit erhöhen. Inzwischen wird sogar eine MNP-Klasse 7 angeboten, die den Datendurchsatz verdreifacht.

Die CCITT-Norm V.42 ist - wie gesagt - kompatibel bis zur MNP-Klasse 4. Die V.42-Modems arbeiten untereinander aber nicht mit dem MNP-Standard, sondern mit dem vom amerikanischen Kommunikationskonzern AT&T vorgeschlagenen Protokoll LAP-M (Link Access Protocol/ Procedure for Modems). Bei der Normierung der Datenkompression verzichtete die Genfer UNO-Behörde völlig auf eine Kompatibilität mit dem Industriestandard MNP. Vielmehr wurde bei der Norm V.42bis auf den „Ziv-Lempel"-Algorithmus gesetzt, der den Computeranwendern von verschiedenen Packer-Programmen her bekannt ist. Modems dieser Norm erreichen im Vergleich zu MNP 5 eine etwa doppelt so hohe Datenkompression. Außerdem werden von V.42bis-Modem nicht komprimierbare Daten erkannt. Bei MNP versucht das Modem etwa, bereits softwaremäßig „gepackte" noch einmal zu komprimieren. Ergebnis: Die Übertragungsgeschwindigkeit wird nicht erhöht, sondern es dauert sogar länger im Vergleich zur einfachen Datenübertragung.

Fazit

Die Entscheidung, welches Modem man anschaffen bzw. ob man nicht doch einen Akustikkoppler kaufen sollte, hängt davon ab, welche Aufgaben man mit dem Gerät erledigen will. Computer-Anwender, die nur sporadisch Daten übers Telefon übertragen wollen und etwa mit einem Laptop unterwegs sind, sollten den Kauf eines Akustikkopplers erwägen. Kleine Pocket-Modems machen aber den Akustikkopplern auch beim mobilen Einsatz Konkurrenz.

Alle anderen DFÜ-Einsteiger sollten sich aber auf dem Modem-Markt umschauen, um sich die faszinierende Welt der Datenfernübertragung zu erschließen. Postzugelassene Hayes-kompatible Modems mit einer Geschwindigkeit von 2400 Baud gibt es heute schon unter 500 Mark zu kaufen. Modems ohne ZZF-Zulassung sind deutlich billiger, doch sollte jeder Computer-Anwender bedenken, daß diese Geräte nach den Bestimmungen der Post (noch) nicht am Netz der Telekom betriebenwerden dürfen, obwohl es ganz einfach ist, die Kabel dieser Geräte mit dem Telefonnetz zu verbinden. Wer ganz tief in die Datenfernübertragung einsteigen möchte, sollte sich auch die komfortableren Geräte mit hohen Übertragungsgeschwindigkeiten, Fehlerkorrektur oder Datenkomprimierung anschauen. Hier gilt allerdings eine alte Weisheit. Luxus hat seinen Preis. Inzwischen gibt es auch für den Atari Programme, mit denen man direkt vom Computer aus ein Fax absetzen kann. Wer sich zu diesem Zweck ein Modem anschaffen möchte, muß darauf achten, daß das Modem auch den notwendigen Chip für das Fax-Senden (bzw. Fax-Empfangen) besitzt.

In der nächsten Folge stellen wir an einem konkreten Beispiel das Angebot einer Mailbox vor und berichten, welche verschiedenen Mailbox-Systeme es im deutschsprachigen Raum gibt.

Christoph Dernbach/Bernhard Krönung



Trouble mit HARDCOPY...

...vor einiger Zeit stürzte ich mich regelrecht ins Choas. Ich ließ Apfelmännchen, Feigenbäume, Juliamengen und Attraktoren mit Hilfe von GFA-Basic V3.xx berechnen. Als diese wunderschönen Grafiken auf meinem Bildschirm flimmerten, kam mir die Idee, daß sich mein Drucker wieder einmal nützlich machen könnte. Ein Blick ins Handbuch und schon hatte ich den Befehl HARDCOPY entdeckt. Jetzt war es kein Problem mehr ihn ins Programm einzubauen. Das Ergebnis kennt jeder. Nun, als Schüler benutzt man natürlich auch PD-Programme, doch ca. 50% der PD-Programm Autoren hatten das selbe Problem wie ich (z. B. bei mehreren Funktionsplottern). Ein Tag später hatte ich eine eigene kleine Druckertreiber Routine für meinen Epson kompatiblen 9-Nadel Drucker STAR LC-10 geschrieben. Um die Qualität von meinen nächsten PD-Programmen zu erhöhen, möchte ich diese kleine „Procedure" an die Allgemeinheit weitergeben:

Matthias Brust u. Christian Roth

```
' ********************************************
' * Von -Matthias Brust- und -Christian Roth- *
' * in GFA Basic 3.xx                        *
' ********************************************
DO
  WHILE GEMDOS(17)=0      ! Drucker empfangsbereit
    ALERT 1,"|Papier einlegen und|Drucker einschalten",1,
                                  "Nochmal|Abbruch",v&
    IF v&=2
      EDIT
    ENDIF
  WEND
  bild$=SPACE$(32034)     ! Speicher reservieren
  FILESELECT #"Grafik laden","A:\*.*","",bild$
  IF bild$=""             ! Sicherheitsabfrage
    EDIT
  ENDIF
  BLOAD bild$,V:bild$     ! Bild laden
  bild$=RIGHT$(bild$,32000) ! Bild formatieren
  adr_bild%=V:bild$       ! Adresse von Bild in Variable
  BMOVE adr_bild%,XBIOS(2),32000  ! auf Bildschirm zeigen
  @druck(adr_bild%)
LOOP
 PROCEDURE druck(adr%)            ! Druckertreiber
  grafik$=CHR$(27)+"*"+CHR$(5)+CHR$(144)+CHR$(1)
  LPRINT CHR$(27);"A";CHR$(8);
  spalte$=STRING$(400,0)
  FOR s%=adr% TO adr%+79
    stop$=INKEY$
    EXIT IF stop$=CHR$(27)
    adr_spalte%=V:spalte$
    ziel%=s%+399*80
    FOR m%=adr_spalte% TO adr_spalte%+399
      POKE m%,PEEK(ziel%)
      SUB ziel%,80
    NEXT m%
    LPRINT grafik$;spalte$
  NEXT s%
RETURN
```

HD-Laufwerke am STE und TT

Zwar wurden die Rechner der Mega STE- und TT-Serie standardmäßig nicht mit HD-Laufwerken ausgerüstet, aber es ist leicht möglich, dies nachzuholen. Beide Computer sind nämlich bereits ab Werk mit einer Schaltung ausgerüstet, die es erlaubt, den Floppy-Controller WD1772 mit 16 MHz zu takten und so HD-Disketten zu einzusetzen. Im Gegensatz zum ST muß die entsprechende Hardware also nicht im nachhinein eingebaut zu werden.

Wird ein externes (oder auch internes) HD-Laufwerk (z.B. TEAC FD235 HF) angeschlossen, so lassen sich HD-Disketten lesen und beschreiben. Ein wenig problematisch ist lediglich das Formatieren solcher Disketten. Unter der Festplatte beider Geräte befinden sich DIP-Schalter, von denen der fünfte auf ON gesetzt werden muß. Anschließend findet man im Dialog zum Formatieren von Disketten einen Knopf mit der Bezeichnung „Hohe Schreibdichte". Ist man nun stolzer Besitzer eines MegaSTE, so steht der Nutzung von HD-Disketten nichts mehr im Wege.

Ist man im Besitz eines TT mit TOS 3.01, so taucht ein ärgerliches Problem auf: Der Knopf „Hohe Schreibdichte" läßt sich aufgrund eines Fehlers nicht anwählen. Dies ist erst ab TOS 3.05 möglich.

Wie kommt man nun trotzdem zu formatierten HD-Disketten? Falls man Zugang zu IBM-kompatiblen ATs hat, bietet es sich an, sich dort einige Disketten auf Vorrat zu formatieren. Nachteil: Solche Disketten zeichnen sich nicht gerade durch eine hohe Datenübertragungsrate aus. Schnelle HD-Disketten auch auf dem TT erhält man, wenn man zum Formatieren Programme heranzieht, die das Formatieren von HD-Disketten unabhängig vom TOS erlauben. Hier bietet sich z.B. das DISKUS-Disk-utility an.

Noch ein wichtiger Hinweis: Werden auf dem TT Programme verwendet, die das ROM ins TT-RAM verlagern (z.B. ROMSPEED oder ROMRAM), kann es vorkommen, das TOS Fehler beim Schreiben auf HD-Disketten meldet. Vemutlich handelt es sich um Timing-Probleme. Hier hilft nur, beim Arbeiten mit HD-Disketten auf die genannten Programme zu verzichten.

US

Haben auch Sie einen *Quick-Tip*?

Standen Sie auch einmal vor einem kleinen, aber schier unlösbarem Problem? Dann, durch Zufall bekamen Sie einen Tip und schon war es gelöst.

Ähnlich haben wir uns diese neue Rubrik in der ST Computer vorgestellt. Aufgerufen sind auch Sie, liebe Leser(innen)! Geben Sie Ihre Erfahrungen weiter, egal, ob es um Anwendungen, Programmieren o.ä. geht.

Wir sammeln Ihre (und unsere) Tips und stellen Sie ggf. in den Quick-Tips vor.

Einsendungen an:

MAXON Computer
ST Computer Redaktion
Stichwort: Quick-Tip
Industriestr. 26
W-6236 Eschborn

Korrektur eines Fehlers bei Wordplus Version 3.15

Die Version 3.15 von Wordplus stürzt ab, wenn beim *Neu Formatieren* das zu trennende Word mit einem Hochkomma (') anfängt. Ich stelle hier eine Möglichkeit vor, wie man diesen Fehler beheben kann. In der Datei WORDPLUS.PRG muß das Byte an der Position $1EBEE von $6C in $64 geändert werden. Die Änderung kann mit einem Diskmonitor vorgenommen werden. In Bild 1 sind noch ein paar Byte davor reassembliert. Falls bei anderen Versionen von Wordplus der gleiche Fehler auftritt, kann man die entsprechende Byte-Sequenz im Programm suchen. Dabei sollte es reichen, wenn man die letzten 6 Byte suchen läßt.

Was wird geändert? An dieser Stelle werden erlaubte Trennstellen im Wort nach Silben gesucht. Über den ersten Buchstaben wird ein Pointer errechnet. Da das Word vorher bereits in Kleinbuchstaben umgewandelt wurde, liegen die erlaubten Werte zwischen $61 (a) und $7A (z). Das Hochkomma ($27) ergibt nach der Subtraktion einen negativen Wert (-58), wenn wir das Ergebnis als signed ansehen und ist damit kleiner als $1A. Interpretieren wir die Zahl nun aber als unsigned, so ist es 65420 und damit größer.

```
Pos.    Inhalt         Reassembliert

1EBCE   4E56 FFF6      LINK      A6,#-$A
1EBD2   426E FFFA      CLR.W     -6(A6)
1EBD6   206E 0008      MOVEA.L   8(A6),A0
1EBDA   4240           CLR.W     D0
1EBDC   1010           MOVE.B    (A0),D0
1EBDE   3D40 FFF8      MOVE.W    D0,-8(A6)
1EBE2   0440 0061      SUBI.W    #$61,D0
1EBE6   3D40 FFF6      MOVE.W    D0,-$A(A6)
1EBEA   0C40 001A      CMPI.W    #$1A,D0
1EBEE   6C72           BGE.S     $72(PC)

Neu: unsigned statt signed
1EBEE   6472           BCC.S     $72(PC)
```

Georg Scheibler, W-4920 Lemgo

Unbekannte Omikron.BASIC-Befehle

Wenn Sie mal in der GEM-Library stöbern, werden Ihnen zwei Befehle auffallen, die weder in Bedienungsanleitung noch in Lehrbüchern zu Omikrons BASIC dokumentiert sind. Es sind dies OUTLINE ON und OUTLINE OFF mit den entsprechenden VDI-Aufrufen *vsf_perimeter(0)* bzw. *vsf_perimeter(1)*. Es ist also doch möglich, die Flächenumrandung direkt im BASIC umzuschalten. Das folgende kleine Programm soll dies demonstrieren:

Dirk Hagedorn, W-4796 Salzkotten 6

```
100 CLS
110 PRINT CHR$(27)+"f";
120 FILL STYLE = 2,1
130 '
140 OUTLINE OFF
150 PBOX 100,100,100,100
160 '
170 OUTLINE ON : 'dies ist die Standardeinstellung
180 PBOX 300,100,100,100
190 '
200 END
```

Drehen als Grundfunktion

Jeder der schon mal versucht hat sein eigenes Zeichenprogramm in BASIC zu schreiben, wird sich überlegt haben, welche Grundfunktionen in sein Programm kommen sollten.

Nun, da es jedem Ziel sein sollte, den Standard zu verbessern, darf man wohl sagen, daß eine 90°-Drehfunktion, in jedes noch so kleine, PD-Zeichenprogramm gehört. Auch wenn man GFA-BASIC für noch so schnell bezeichnen kann, dauerte mein Benchmark 3.775 s, in meiner Assembler-Lösung, die in GFA-BASIC mit Parameterübergabe eingebunden wird, benötigt man"nur" noch 0.725 s. Also fast viermal so schnell, welches für Assembler eigentlich noch langsam ist. Natürlich könnte man die Geschwindigkeit noch um einiges steigern, wenn man nicht pixelweise bzw. bitweise, sondern wordweise vorgehen würde. Also 16 Pixel auf einmal im Speicher auswerten, und erst dann darstellen täte.

Nur diese Vorgehensweise müßte schon etwas genauer erklärt werden, um den internen Kontext zu verstehen. Also nichts mehr für die Quick-Tips.

Matthias Brust und Christian Roth

```
' Geschrieben von -Matthias Brust- und -Christian Roth-'
' --GFA Basic 3.xx--
INLINE start%,200        ! BASIC-interne Speicherreservierung
BLOAD "90_GRAD.O",start% ! Laden
DEFFILL 1,4,2            ! Für kleine Grafik...
x1&=0
y1&=0
x2&=190
y2&=190
PBOX x1&,y1&,x2&,y2&              ! ...bis hierher.
~C:start%(x1&,y1&,x2&,y2&)        ! Maschinen Programm aufrufen
'
' --GFA Basic 2.xx--
DIM feld%(200/4)                          ! Feld einrichten
start%=VARPTR(feld%(0))                   ! Adresse holen
BLOAD "90_GRAD.O",start%
DEFFILL 1,4,2
x1%=0
y1%=0
x2%=190
y2%=190
PBOX x1%,y1%,x2%,y2%
VOID C:start%(x1%,y1%,x2%,y2%)
```

```
; ********************************************
; * Von -Matthias Brust- und -Christian Roth- *
; * in GFA Assembler V1.5                    *
; ********************************************
intin       equ 8                          ; wichtige Adressen
ptsin       equ 12
lstlin      equ 32
            .MACRO lineainit               ; Init Macro
            .DC.w $a000                    ; Adresse von Init
            movea.l   a0,a5                ; Zeiger retten
            move.w    #-1,lstlin(a5)       ; Sollte -1 sein
            .ENDM
            .MACRO putpixel farbe,x,y      ; entspricht PSET
            movea.l   intin(a5),a0
            move.w    \1,(a0)
            movea.l   ptsin(a5),a0
            move.w    \2,(a0)
            move.w    \3,2(a0)
            .DC.w $a001
            .ENDM
            .MACRO getpixel x,y            ; Entspricht PTST
            movea.l   ptsin(a5),a0
            move.w    \1,(a0)
            move.w    \2,2(a0)
            .DC.w $a002
            .ENDM
code:       move.w    4(sp),d3      ; Parameter x1&
            move.w    6(sp),d4      ; Parameter y1&
            move.w    8(sp),d5      ; Parameter x2&
            move.w    10(sp),d6     ; Parameter y2&
            move.w    d5,d7         ; d5 duplizieren
            lineainit
loop1:      move.w    d7,d5         ; Zähler neu setzen
loop2:      getpixel  d5,d6
            cmpi.w    #1,d0         ; Pixel gesetzt?
            blt       weiter        ; Nein -> weiter
            putpixel  #1,d6,d5      ; Ja -> setzen
weiter:     cmp.w     d3,d5
            dbls      d5,loop2      ; x Koord. Schleife
            cmp.w     d4,d6
            dbls      d6,loop1      ; y Koord. Schleife
            rts
```





ARACHNOPHOBIA - ein Arcade-Adventure aus dem Hause Titus

Titus wird demnächst das Arcade-Adventure ARACHNOPHOBIA auf den Markt bringen, das es in sich haben soll. Das erste Demo läßt zumindest hoffen und verspricht eine Menge Spaß.

Hallo Spiele-Fans,



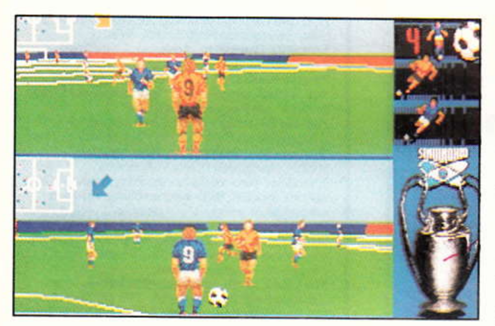
BRIDES OF DRACULA greift die Story vom blutsaugenden Vampir auf.

BRIDES OF DRACULA
Das Horrorspielchen kann man ein wenig mit dem C64-Klassiker Jet Set Willy vergleichen, allerdings sind Grafik und Sound wesentlich besser. Die Handlung ist einigermaßen okay und verspricht einigen Spielspaß.



OUTZONE - Action vom Feinsten

Bei OUTZONES handelt es sich um ein Ballerspiel, in dem Sie verschollene Raumschiffe aus den gefährlichen OUTZONES retten müssen. Sieben verschiedene Welten zu je 20 Bildschirmbreiten, Parallax-Scrolling, 70 unterschiedliche Monster, 45 verschiedene Farben, 50 Bilder pro Sekunde und vieles mehr zeichnen OUTZONE aus.



I PLAY 3D SOCCER - eine 3D-Fußballvariante

I PLAY 3D SOCCER
Seit der Fußballweltmeisterschaft kommen immer mehr Fußballsimulationen auf den Markt. Die neueste Variante ist 3D SOCCER, welches eben eine echte 3D-Perspektive liefert. Beispielsweise werden die Spielfiguren beim Entfernen kleiner und beim Näherkommen größer.



Im Sherwood Forest geht's heiß her.

Auf den ersten Blick ähnelt ROBIN HOOD ein wenig dem Spiel Populous. Aber der Schein trügt, vielmehr handelt es sich bei ROBIN HOOD um ein sehr komplexes Action-Adventure, das den Spieler in den Sherwood Forest führt. Der Spieler kann mit allen Charakteren in Kontakt treten, Informationen erfahren oder Waren tauschen.



RECTANGLE glänzt durch ein einfaches Spielprinzip.

In diesen Tagen kommt RECTANGLE von Turtle Bytes auf den Markt. Bei diesem Spiel geht es darum, vorgefertigte Figuren durch Quadrate zu Rechtecken zu formen. Die Figuren fallen herab und unser Held schießt die Quadrate auf die Figuren. Ist ein Rechteck geformt, verschwindet es. Die Spielidee von RECTANGLE ist einfach, trotzdem hat das Spiel einen hohen Reiz und erfordert eine gehörige Portion Strategie.



Nette Grafik und ein feines Gameplay sollen MAGIC POCKETS auszeichnen.

MAGIC POCKETS
Der Spieler schlüpft in die Rolle des kleinen Bitmap Kid, der sich durch insgesamt fünf Levels angefüllt mit zahlreichen Widersachern, kämpfen muß. Das Action-Spiel glänzt mit netter Grafik und einem sehr guten Scrolling.

Super Cars II



Bei „Super Cars II“ hat man endlich doppelt soviel Spaß fürs Geld: jetzt können nämlich zwei PS-Protzer gleichzeitig losrasen. Moment! Doch nicht sofort! - Erstmal müssen da ein paar Features eingestellt werden, zum Beispiel der Ein- oder Zwei-Spieler-Modus. Dann wird's kritisch: welche von drei Schwierigkeitsstufen soll es sein? In Stufe „easy“ ist die Straße mustergültig, frei von Schlaglöchern und recht gut zu übersehen. Anders bei „hard“, da treiben stockfinstere Tunnel, chaotische Verkehrsführung und glitschige Fahrbahndecke den Fahrer zur Verzweiflung. Bei „medium“ verhält es sich wie beim Steak: es geht ziemlich roh zu. Als Gaspedal kann entweder der Feuerknopf des Joysticks dienen, oder man belegt den Knopf so, daß er zum Bremsen benutzt werden kann. „Plöps!“ Ach - das war das Startsignal! Die neun anderen Rennfahrer geben schon ordentlich Gas. Dann geht's in wilder Jagd über die Pisten und in die Kurven. Steht die Schadensanzeige fast auf Null, und/oder hat man sich am Ende des Rennens nicht unter den ersten Fünf plaziert, ist das Rennen vorbei. Schade, denn auf jedem Schwierigkeitslevel hat Hersteller Gremlin sieben unterschiedliche Routen vorgegeben, bei denen eine verzwickter ist als die andere. Es gibt vertrackte Haarnadelkurven und lange, nachtfinstere Tunnelstrecken, in denen man genauso gut mit geschlossenen Augen fahren könnte. Und wer hätte mitten auf einer zivilisierten Rennstrecke eine Sprungschanze vermutet? Sogar, wenn man mit „easy“ anfängt, plagen einen sehr bald die bösen Ideen der Programmierer. Im härtesten Modus passiert es dann alle paar Meter, daß der eigene Wagen an die Leitplanke scheppert, gerammt oder von den üblen Raketen getroffen wird. Wenn man dabei nur Zeit und Schadenspunkte verlöre, ginge es ja noch - aber es kostet auch noch Geld. Also trabt der Spieler zwischen den Rennen los und versucht, preisgünstige



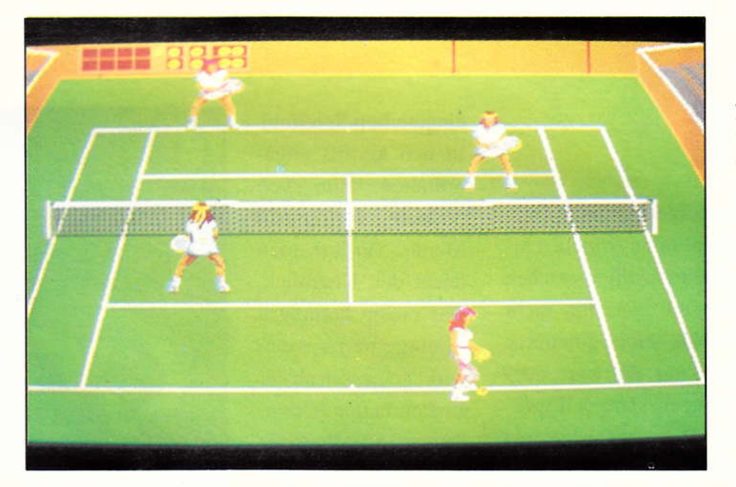
Ersatzteile zu erwerben, die man entweder für das eigene Fahrzeug verwendet oder aufhebt. Ohne eine Ladung Kriegsgerät im Wagen kann man sich den Sieg fast abschminken. Martialische Raketen, die nach vorn oder hinten abgeschossen werden, liegen in den Ladenregalen. Sanfte Gemüter können sich zwischendurch beim Rätseln erholen. Was bedeutet dieses oder jenes Verkehrsschild? Jeder müßte es wissen, und wer es weiß, bekommt zur Belohnung ein kleines Taschengeld. Natürlich stellen die naseweisen Typen auch schwierigere Fragen. „Super Cars II“ wird dadurch zumindest vielseitiger. Musikbegleitung während des Rennens fehlt. Bleibt noch, das dezent wackelige Scrolling zu bemängeln. Und das waren auch schon alle Nachteile. „Super Cars II“ ist so abwechslungsreich, spannend und rasant, daß es glatt an seinem Vorgänger vorbeizieht. Leute, die digitale Asphaltabenteuer lieben, sollten gleich in den nächsten Laden stürmen und sich Super Cars II zulegen.

CBO

Great Courts II



Für Fans von fliegenden Filzbällen hat die Mülheimer Spieleschmiede „Blue Byte“ ein schmackhaftes Bonbon zu bieten - die Fortsetzung des Tennishits „Great Courts“. Jeder Mitspieler bestimmt, wo seine Stärken liegen sollen. Anhand von Skalen zwischen 1 und 100 bestimmt man seine Fähigkeiten bei Aufschlag, Schmetterball, Volley, Vorhand und Rückhand. Eine weitere Anzeige informiert über die Kondition. Wie im klassischen Rollenspiel ordnet man seinem Tenniscrack Charakterpunkte für individuelle Eigenschaften zu. Schließlich belegt jeder Tennisfreak einen Joystick-Ausgang. Wenn mehr als zwei Spieler mitmachen, löst ein handelsüblicher Vier- Player-Adapter das Problem mit den Ports. Dann geht es noch um die Wahl des Spielmodus'. Für Neulinge eignet sich der sogenannte Junior-Mode, da hier die Beinarbeit softwaregesteuert ist und der Spieler seine ganze Aufmerksamkeit den Ballkontakten widmen kann. Wer sich sicher fühlt, wählt den Normal-Mode und rennt selber. Sinnvollerweise beginnt man das Spiel im Trainingsmodus. Über ein Preferences-Menü sucht man sich unter Hartplatz, Sandboden oder Rasen den gewünschten Spielgrund aus und bestimmt, wieviele Gewinnsätze es geben soll. Schließlich fragt der Computer, ob sich zwei, drei oder vier Balljäger auf dem Court tummeln sollen. Unverzichtbar für Anfänger: die Ballmaschine. Im speziellen Menü lassen sich individuelle Trainingseinheiten mit dem mechanischen Übungspartner zusammenstellen. Daß sich Charakterzüge und Spielstärke im Turniermodus während des Spiels verändern, zeigt, wie



wirklichkeitsnah „Great Courts II“ umgesetzt wurde. Dieses Feature verlangt vom Spieler, auf seine Gesundheit zu achten, an seiner Schlagtechnik zu arbeiten und mit den Kräften hauszuhalten. Soweit die konzeptionelle Seite des zweiten „Great Courts“. Wohl durchdacht sind jedoch nicht nur die technischen Einzelheiten der Simulation. Auch die Grafik begeistert. Die Spielfiguren sind detailliert dargestellt. Ihre Bewegungen sind glatt, schnell animiert und annähernd natürlich. Mit grafischen Gags zeigen die Programmierer auch noch, wie witzig simuliertes Tennis sein kann. Man richte beispielsweise beim Aufschlag einer weiblichen Spielerin seinen Augenmerk auf das hochflatternde Tennisröckchen. Situationsgerechte Sounds schaffen dazu Spannung und viel Atmosphäre. „Great Courts II“ ist tatsächlich noch aufregender als die Erstausgabe.

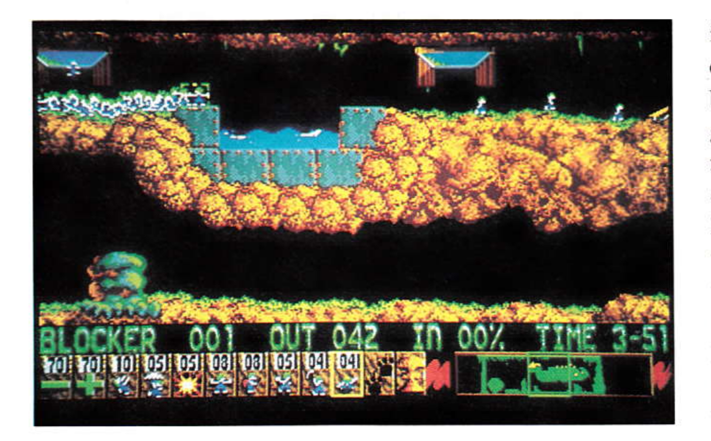
CBO

Lemmings



Lemmings sind kleine, arktische Nagetiere, welche in diesem brandneuen Spiel von Psygnosis die Tendenz haben, sich auf irgendeine Art und Weise, meistens jedoch durch Sturz aus großer Höhe, das Leben zu nehmen. Auf der „European Trade Show" wurde dieses Spiel kürzlich von einigen sehr bekannten Magazinen zum absolut besten Spiel des Jahres gewählt. Wirft man ein Blick auf die Grafiken, erfolgt zunächst eine Ernüchterung. Kleine leblos erscheinende Sprites, die sich langsam bewegen. Soll das ein Psygnosys-Spiel sein? Was sich aber dahinter verbirgt, ist tatsächlich eines der am besten spielbaren, herausfordernsten und originellsten Spiele seit langer, langer Zeit. Das Programm ist so fesselnd, daß die etwas nüchterne Grafik nebensächlich wird, sobald man beginnt es zu spielen.

Das Ziel des Spieles ist einfach umrissen. Es existieren über 100 Level, jeder ist völlig verschieden aufgebaut, und der Spieler muß versuchen einen gewissen Prozentsatz der auftauchenden Lemmings vor dem sicheren Tod zu bewahren. Das erste Level ist noch sehr einfach, aber der Schwierigkeitsgrad steigt kontinuierlich mit höheren Levels an. Vor jedem neuen Bild bekommt der Spieler gezeigt, wieviel Lemmings insgesamt auftauchen werden, und wieviel Prozent er davon mindestens durchbringen muß, um zum nächsten Level zugelassen zu werden. Die Lemmings suchen, sobald sie auftauchen, den nächsten Weg zum tiefen Abgrund, lassen sich über einem Feuer rösten oder laufen anderen tödlichen Gefahren blind entgegen. Rettend eingreifen kann der Spieler,



indem er einzelnen Lemmings besondere Eigenschaften gibt. Z.B. um den Strom von Lemmings umzuleiten, damit sie nicht alle den Abgrund hinunterfallen, braucht nur der Lemming, der vorneweg läuft in einen „Blocker" umgewandelt zu werden. Er stellt sich dann mit ausgebreiteten Armen hin, und alle anderen, die bis zu ihm vordringen, drehen einfach wieder um. Ebenso kann man „Brückenbauer" erzeugen, oder Lemmings, welche permanent Löcher in den Boden graben. Das alles hat nur den einen Sinn, die kleinen Nager so geschickt zu leiten, daß sie den Ausgang, ein kleines Häuschen mit wehender Fahne, erreichen. Die Anzahl der zu vergebenden Extras ist nicht unbegrenzt. Man wird sehr schnell dazu verleitet, einfach allen Lemmings z.B. einen Fallschirm zu verpassen, damit sie einen Sturz aus großer Höhe überleben. Aber was, wenn alle Fallschirme weg sind? Auch die Brückenbauer können nicht unbegrenzt ihrer Maurertätigkeit nachgehen. Sie haben nur zwölf Planken zur Verfügung; ist die 12. gelegt und der Spieler vergißt, den Lemming erneut als Brückenbauer einzusetzen, ist sein Fall unvermeidbar. Ein Blocker ist ebenso verloren, denn er kann nicht wieder dazu bewegt werden, weiterzugehen. Beseitigen kann man ihn nur durch eine Sprengung, die er natürlich nicht überlebt. Kurzum, man sollte sich genau überlegen, welche Eigenschaften man welchem Lemming zuteilt, damit auch genügend übrig bleiben, um den geforderten Prozentsatz zu erfüllen. Einige von ihnen wird man opfern müssen, um vielen anderen das Leben zu retten. Um ein Level zu lösen, sollte man sich also zunächst über den richtigen Lösungsweg im Klaren sein. Unendlich viel Zeit hat man allerdings nicht. Wie bei vielen Spielen, läuft auch hier die Uhr gegen den Spieler. Einige Level sind so vertrackt, daß man Stunden damit zubringt, um den Lösungsweg zu finden, und dann, auf einmal, quasi per Geistesblitz ist alles klar. Hilfreich ist, daß der Spieler nach jedem Level einen Geheimcode bekommt, um später bei diesem wieder einsteigen zu können.

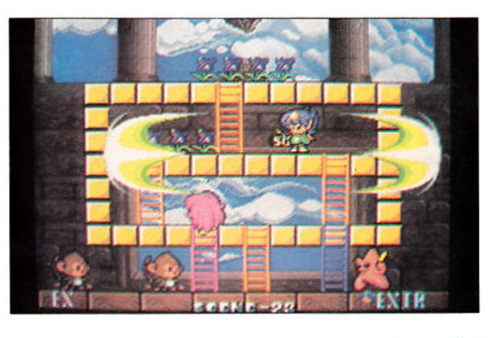
Lemmings kann frustrierend und erfrischend zugleich sein. Manchmal sind wirklich nur gut durchdachte Lösungen möglich, dann wieder sieht der Spieler auf den ersten Blick den richtigen Weg. Wenn man ein Spiel sucht, daß jemanden die nächsten sechs Monate voll beschäftigen kann, dann ist Lemmings genau das richtige. Es könnte ein Kultspiel werden. Angekündigt sind weitere Level-Disketten und ein Construction-Set. Eine Version mit wesentlich verbesserter Grafik soll zudem folgen.

ddf/cm

ROD-LAND...



... präsentiert sich im selben Stil wie der Klassiker Rainbow Island: einfaches Spielprinzip mit vielen niedlichen Charakteren in insgesamt 33 Levels. Die Aufgabe von Tam und Rit, den beiden Helden von ROD-LAND, besteht darin, ihre entführte Mutter aus dem Maboots Tower zu befreien. Die einzelnen Spielabschnitte bestehen aus einem einzigen Bildschirm, in ihnen befinden sich zahlreiche, mitunter unterschiedliche Monster und Blumenbeete. Die Blumen müssen gepflückt und die Widersacher ausgeschaltet werden. Glücklicherweise besitzen Tam und Rit einen Zauberstab, der Leitern herbeizaubern kann, und ein magisches Lasso, mit dessen Hilfe die Widersacher, sind sie darin gefangen, durch mehrmaliges Aufdotzen unschädlich gemacht werden können. Weiterhin existieren 9 Oberfieslinge, die es wirklich in sich haben. Natürlich gibt es zahlreiche Extras und versteckte Räume, die das Spiel weiter aufwerten. ROD-LAND ist ein Spiel, das süchtig macht und einen nicht mehr losläßt. Die



Grafik ist zwar nicht der allerletzte Schrei, kann aber trotzdem gefallen. FX und Musik sind angemessen und überzeugend.

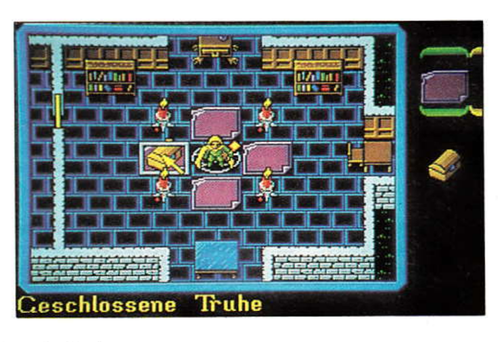
ddf

LORDS OF CHAOS



Das erfrischende Fantasy- und Strategie-Rollenspiel LORDS OF CHAOS kombiniert Ideen aus ULTIMA, DUNGEON MASTER und SSI-Produkte. Am Anfang kann man sich eine eigene Gruppe zusammenstellen, die sich zahlreichen Aufgaben gegenübersieht. LORDS OF CHAOS gefällt, das liegt an dem sauberen Konzept, dem langsam steigenden Schwierigkeitsgrad und der deutschen Benutzerführung. Die Grafik und die Animationen holen zwar nicht viel aus dem AMIGA heraus; aber hier zählt mehr das Gameplay. Adventure-Experten werden über manche zu lösende Aufgabe nur müde lächeln, aus diesem Grund eignet sich das Spiel besonders für Anfänger.

ddf



JAHANGIR KHAN WORLD CHAMPIONSHIP SQUASH



Wie Sie vielleicht wissen, ist Squash ein sehr schneller Sport, nicht anders verhält es sich bei JKWCS. Es stehen verschiedene Spielmodi zur Verfügung; beispielsweise kann man an einem kompletten Turnier teilnehmen oder nur ein einfaches Spielchen gegen den Computer machen. Bevor man aber die ersten Erfolgserlebnisse erzielt, muß man schon ein wenig üben. Ist man aber erst einmal eingespielt, macht JKWCS einen Heidenspaß. Die Grafik ist zwar nicht die allerbeste, man kann aber mit ihr auskommen. Auch der Sound beschränkt sich nur auf Ballgeräusche, was meiner Meinung nach genügt. Insgesamt betrachtet kann man die Squash-Simulation allen sportbegeisterten ST-Anwendern empfehlen.

ddf

HYDRA



Die Firmen Domark/Tengen sind Spezialisten in Sachen Automatenkonvertierung. Das merkt man auch bei HYDRA. Die Rennsimulation versetzt den Spieler aufs Wasser, wo es ganz schön zur Sache geht. Das Rennboot ist mit zahlreichen Extras ausgestattet, beispielsweise mit einer Bordkanone, mit deren Hilfe man Hindernisse oder gegnerische Kapitäne aus dem Weg räumen kann. Leider bietet HYDRA nicht sehr viel Abwechslung, so daß man bereits nach kurzer Zeit das Handtuch bzw. den Joystick in die Ecke wirft. Grafik, Sound und Spielspaß sind bestenfalls Durchschnitt.

ddf





MOONSHINE RACERS



Millennium präsentiert mit MOONSHINE RACERS ein eigenwilles Autorennspiel, das einen in den rauhen Westen von Amerika führt. Billy Joe hat sich zur Aufgabe gemacht, verschiedene Bars zu beliefern; und da die Barbesucher durstig sind, muß die Lieferung schnell vonstatten gehen. Allerdings hat der Sheriff etwas gegen zu schnelles Fahren. MOONSHINE RACERS ist ein typisches Rennspiel mit einigen Extras und mit viel Witz. Hat man die Ware abgeliefert, kann man gegen Bares seinem alten Laster etwas mehr Power in Form von Turboladern oder stärkeren Motoren verpassen. Die Grafik ist nett und die Animation recht flüssig.

ddf

Parametrische Prozeduren und Funktionen

in ST-Pascal

Vor nicht allzu langer Zeit las ich auf Seite 7-2 meines Handbuches zu ST-Pascal-Plus, Version 2.0x den Satz »Der ISO-Standard wird von Pascal plus vollkommen erfüllt...« „Aha", dachte ich. »Geh nach Rom, Cäsar Bericht erstatten. Sag ihm: Ganz Gallien ist besetzt! Dann wird er dich fragen: Ganz? Du antwortest ihm: Ganz! Und er wird verstehen«[1]

Hastig blätterte ich weiter zu jenen Handbuchgefilden, die ihr kärglich Dasein abseits jeder Beachtung fristen (wer liest schon gerne die *n+1*. Allgemeine Einführung in Pascal?): Seiten 7- 59ff, »Vereinbarung von formalen Parametern«. Und wirklich, in keiner Ankündigung stand es: Ab der Version 2.00 gestattet ST-Pascal die Übergabe von Prozedur- und Funktionsnamen als Parameter von Unterprogrammen. Sie können also einem Unterprogramm beim Aufruf Prozedur- oder Funktionsnamen übergeben, genau so, wie Sie auch Variablen oder Konstanten übergeben.

Wozu soll das gut sein?

... höre ich die Philister und die Defätisten rufen und doch ziehe so mancher Münchhausen an diesem Schopfe sich aus dem Programmsumpf. Wie hängen Sie denn Ihre Pascal-Prozeduren in die VBL-Slots? Gar nicht, Ihnen fehlt die Prozeduradresse? Eben! (...ein breites Grinsen bemächtigt sich des Autors beim Tippen dieser Zeilen...)

Aber im Ernst: Die „Parametrischen Prozeduren und Funktionen", wie Niklaus Wirth sie nennt, verbessern vor allem die Möglichkeiten der modularen Programmierung. Wenn Sie beim Programmeschreiben jedesmal das Rad neu erfinden und keine externen Unterprogramme benutzen, könnte Ihnen die Möglichkeit der Übergabe von Unterprogrammnamen an andere Unterprogramme eigentlich egal sein. Wenn Sie sich aber gerne Unterprogramme schreiben, um diese dann in verschiedenen Programmen zu verwenden, sieht die Sache schon anders aus. Niklaus Wirth gibt in einem sehr empfehlenswerten Buch ein Anwendungsbeispiel[2] aus dem Gebiet der Mathematik: Er beschreibt eine Funktion *Simpson*, die für eine beliebige einstellige Funktion *f(x)* das bestimmte Integral zwischen den Grenzen *a* und *b* nach dem entsprechenden Algorithmus berechnen soll. Da die Funktion *f(x)* einerseits innerhalb von *Simpson* mit verschiedenen Werten berechnet werden muß, andererseits von Aufruf zu Aufruf Simpsons sich unterscheiden kann, definiert Wirth seine Funktion einfach mit der formalen Parameterliste *(a, b: real; function f: real)*. Mit der Anweisung *s:= Simpson (0.0, 1.0, g)* würde dann der Variablen *s* der Wert des bestimmten Integrals der Funktion *g* in den Grenzen von 0 bis 1 zugewiesen. Die Funktion *Simpson* ist damit universell einsetzbar.

Fast wie im richtigen Leben

Ach so, Sie integrieren nie in Ihren Programmen. Auch gut. Dann kann ich Sie beruhigen, ich habe mir für Sie zwei Anwendungen direkt aus dem grauen Atari-Alltag ausgedacht, die Sie sicherlich von der Nützlichkeit der Parametrischen Prozeduren überzeugen werden.

Da wäre zuerst eine neue *Get_Event*-Funktion, die Ihnen den Kampf mit den Rechtecklisten beim Wiederherstellen von Fensterinhalten abnimmt. Verwenden Sie diese Funktion in Ihren Programmen, dann können Sie die Ereignismeldung *WM_-Redraw* aus Ihrem Gedächtnis streichen. Die neue Ereignisfunktion reagiert nämlich automatisch auf diese Meldung, veranlaßt den nötigen Redraw und kommt erst wieder aus den AES-Gefilden, wenn etwas wirklich Wichtiges passiert ist. Wie das genau funktioniert, davon künden die Listings 1 und 2. Ich erzähle weiter unten natürlich auch noch etwas darüber.

Das zweite Beispiel (Listings 3 und 4) ist ein wirklicher Leckerbissen für Faustsche Charaktere. Ich verwende nämlich einen durchaus illegalen Trick, um einen Dialog mit einem Objekt vom Typ *G_ProgDef* ausschließlich mit Pascal zu programmieren. Dieser Trick funktioniert natürlich auch wieder nur mit Parametrischen Prozeduren. Er verleiht allerdings die Macht über so eigenwillige Atari-Geister wie die „VBL-Slot-Programme", die Maustastendruck- und Mausbewegungsroutinen des VDI und natürlich auch die *G_Progdef*-Objekte des AES. Den Umgang mit letzteren will ich mit diesem Beispielprogramm auch noch demonstrieren. Da gibt es nämlich bei ST-Pascal-plus ein wenig zu beachten. Doch zuerst die Grundlagen:

Was sind Parametrische Prozeduren?

Ganz einfach: Denken Sie sich ein Programm, in dem Sie die Prozeduren *p*, *q*, *g* und *r* definiert haben. In der Parameterliste von *r* taucht noch die Definition procedure *x* auf. Beispiel:

```
program minitest;

procedure p (i: integer);
begin {...} end;
procedure q (j: integer);
begin {...} end;
procedure g (var k: integer);
begin {...} end;
procedure r ( procedure x
            (k: integer) );
begin {...}; x(99); {...} end;

begin {...};
   r (p);
   r (q);
   {...}
end.
```

Der Parameter *x* ist jetzt eine formale Parametrische Prozedur. Beim ersten Aufruf führt *r* intern die Prozedur *p* aus, beim zweiten die Prozedur *q*. Beide Male mit dem Parameter 99. Was diese beiden Prozeduren letztlich machen, geht die Pro-

zedur *r* nichts an. *X* ist also ein Platzhalter (formaler Parameter) für Prozeduren, genau so, wie es Platzhalter für Variablen und Werte gibt. Die Prozedur *g* kann *r* nicht als Parameter übergeben werden, weil ihre formale Parameterliste aus einem Variablenparameter besteht und nicht aus einem Werteparameter wie bei *x*. Standardprozeduren und -funktionen können übrigens auch nicht als Parametrische Unterprogramme übergeben werden.

Mit Funktionen geht das Ganze natürlich analog. Hier steht dann eine *function*-Deklaration in der Parameterliste von *r*: *function x (k: integer): Funktionstyp*.

Sie können eine Parametrische Prozedur in einem Unterprogramm genau so verwenden, wie Sie eine ganz normale Prozedur verwenden würden: als Anweisung. Eine Parametrische Funktion verwenden Sie dann gerade wie eine normale Funktion: zum Beispiel auf der rechten Seite einer Zuweisung. Im Unterschied zu Variablen- oder Werteparametern können Sie einer Parametrischen Prozedur oder Funktion innerhalb Ihres Unterprogrammes allerdings keine neuen „Werte", also Unterprogrammnamen, zuweisen. Eine Zuweisung *x:= q* in unserem Beispiel wäre also nicht erlaubt!

Pascal-Programm als Klassengesellschaft

Generell können Sie einen Parameter für eine Prozedur oder Funktion in der Parameterliste eines Unterprogrammes definieren, indem Sie einen *kompletten Prozedur- oder Funktionskopf* in die Liste eintragen. Mit diesem Parameter haben Sie dann eine ganze Klasse von Prozeduren oder Funktionen definiert. Solche Klassen unterscheiden sich immer durch ihre Parameterliste (und bei Funktionen zusätzlich durch den Funktionstyp). Die Namen der Parameter sind dabei beliebig wählbar, sie müssen aber in Reihenfolge, Anzahl und Typ exakt übereinstimmen. Die beiden Prozeduren *p* und *q* aus dem Beispiel gehören deshalb der selben Klasse an wie *x*, die Prozedur *g* einer anderen. Für den Platzhalter *x* können Sie jetzt dem Unterprogramm r alle Namen von Prozeduren übergeben, die der gleichen Klasse angehören. Merke: Klassen sind für Parametrische Prozeduren in etwa dasselbe wie Typen für Variablen- oder Wertparameter.

Dieses Unterteilen in Klassen ist übrigens eine Einschränkung gegenüber dem ISO-Standard, der die Angabe der Parameterliste nicht vorsieht - meines Erachtens aber eine sinnvolle. Wie sonst könnte schon zur Compile-Zeit der Fehler entdeckt werden, wenn im Unterprogramm *r* die Anweisung *x(99,98)* stünde, mithin einer Prozedur Parameter übergeben werden, die sie gar nicht verarbeiten kann? Oder was soll die Prozedur *g* mit der Variablen 99 anfangen? Bomben wären die unausweichliche Folge.

Wie sieht es aber mit der Umgebung eines Unterprogrammes aus, das als Parametrisches Unterprogramm in einem anderen Unterprogramm ausgeführt wird? Welche Konstanten, Variablen und Unterprogramme sind ihm bekannt?

Etwas über Umgebungen

Den Prozeduren und Funktionen, die als Parameter übergeben werden, bleibt immer nur die Umgebung sichtbar, in der sie definiert wurden. Somit können sie globale Konstanten, Variablen und Unterprogramme verwenden, auch wenn in dem sie ausführenden Unterprogramm namensgleiche Konstanten, Variablen oder Unterprogramme mit anderen Inhalten definiert wurden. Diese lokalen Deklarationen sind für das parametrische Unterprogramm unsichtbar. Einer Parametrischen Prozedur oder Funktion geht es also genauso wie einer Prozedur/Funktion, die vor einem Unterprogramm definiert wurde, das sie aufruft. Soweit die Grundlagen. Kommen wir nun zu den Anwendungsbeispielen.

Fenster-Redraw automatisch

Listing 1 ist ein Programm-Modul, das die Funktion *AutoRedraw_Event* zur Verfügung stellt. Compilieren Sie dieses Modul und binden Sie den Funktionskopf mit der External-Direktive in Ihre Programme oder noch besser an das Ende der Datei GEMSUBS.PAS ein. Sie können die Funktion dann in Ihren Programmen statt der PASGEM-Funktion *Get_Event* verwenden, wenn Sie das Objektmodul LISTING1.O als zusätzliche Link-Datei angeben.

Die neue Ereignisfunktion hat exakt die gleichen Parameter wie die alte. Zusätzlich übergeben Sie jedoch noch dem Parameter *Draw* eine Prozedur. Dieser Prozedur kommt die Aufgabe zu, den Inhalt des Fensters mit der Kennung *Handle* innerhalb des Rechteckes (Clipping-Bereich) zu zeichnen, das durch die Koordinaten *x*, *y*, *w und h* gegeben ist. Wie die Prozedur

das macht, legen Sie selbst fest. Im Beispielprogramm von Listing 2 habe ich einfach eine Prozedur *FensterDraw* definiert, die in das angegebene Rechteck einen umrandeten Kasten mit einem diagonalen Kreuz zeichnet. Die Prozedur ist der Arbeitsweise des Demoprogrammes WINDOW_TEST von CCD nachgebildet.

Die Funktion *AutoRedraw_Event* verhält sich nun ganz ähnlich wie die alte Funktion *Get_Event*. Der einzige Unterschied besteht darin, daß sie niemals eine Meldung vom Typ *WM_Redraw* zurückgibt. Diese Meldung wird intern verarbeitet. Sie übergeben der neuen Ereignisfunktion also genau dieselben Parameter, die Sie auch der Funktion *Get_Event* übergeben hätten, und erhalten auch die gleichen Werte zurück.

Wird *AutoRedraw_Event* aufgerufen, addiert sie zuerst einmal den Typ *E_Message* auf die Ereignismaske. Sollten Sie an Meldungsereignissen kein Interesse zeigen, ist das auch notwendig, um überhaupt anstehende Redraw-Meldungen zu empfangen. Danach wird in einer Schleife die alte *Get_Event*-Funktion mit Ihren Parametern aufgerufen. Retourniert diese ein Meldungsereignis, prüft *Auto-Redraw_Event*, ob es sich um eine Window-Redraw-Meldung handelt. Ist dies der Fall, dann wird GEM der Anfang eines Updates mitgeteilt, die Maus ausgeschaltet und die komplette Rechteckliste des Redraws abgearbeitet, wobei der jeweilige Überlappungsbereich der Draw-Routine in den Koordinaten *x*, *y*, *w* und *h* übergeben wird. Um die Routine in die Lage zu versetzen, für verschiedene Fenster verschiedene Zeichenarbeiten auszuführen, wird ihr auch noch die Fensterkennung über den Parameter *Handle* mitgeteilt.

Ist die Rechteckliste abgearbeitet, wird die Maus wieder auf den Bildschirm gebracht und das Update-Ende angezeigt. Das Meldungsereignis muß noch aus der temporären Ereignisvariablen ausgeblendet werden, da das Hauptprogramm ja nicht mehr mit solchen Lappalien belastet werden soll.

War die Redraw-Meldung das einzige Ereignis, das von *Get_Event* angezeigt wurde, gibt es keinen Grund, zum Hauptprogramm zurückzukehren. Wir durchlaufen also die Ereignisschleife noch einmal. Lieferte *Get_Event* allerdings (zusätzlich) ein anderes Ereignis, dann sollte das Hauptprogramm davon erfahren, und die Funktion beendet sich an dieser Stelle, nicht ohne das Ereignis zu retournieren.

Das zugehörige Beispielprogramm in Listing 2 ist recht einfach gehalten. Zuerst öffnet es vier Fenster und stellt ein Semaphor (Ampel) entsprechend auf vierfach-Grün ein. In einer Ereignisschleife (*event_loop*) wartet unser Programm darauf, daß kein Fenster mehr offen ist (FensterSemaphor = KeinFenster = Rot). Die Zeit bis dahin vertreibt es sich damit, auf Anwenderaktionen wie Fenster verschieben, an die oberste Position bringen und natürlich Fenster schließen zu reagieren, wobei letzteres die Aktualisierung des Semaphors bedingt. Die Inhalte der Fenster werden ja von der *FensterDraw*-Prozedur auf Anweisung der grauen Eminenz *AutoRedraw_Event* aktualisiert. Selbst das komplette Neuzeichnen gleich nach dem Öffnen eines Fensters wird von der neuen Ereignisfunktion angeregt! Es gibt also keinen (sic!) Unterschied zwischen einem Fenster-*Draw* und einem Fenster-*Redraw*. Den Aufruf der Zeichenprozedur im Beispielprogramm suchen Sie daher vergeblich!

Um die ganze Sache noch etwas abwechslungsreicher zu gestalten, wartet das Programm auch noch auf Tastendrücke, um den Tastaturcode der gedrückten Taste in einer Alertbox zum Gaudium aller Beteiligten in hexadezimaler Schreibweise zu verkünden. Willfährig entlastet uns die *WriteV*-Prozedur dabei von ekler Konvertierarbeit.

Soviel zu diesem Thema. Steigen wir jetzt vom schönen Pascal-Himmel hinab in den Orkus betriebssystemnaher Programmierung...

Pascal joins USRBLK

Vielleicht haben Sie das auch schon erlebt: Sie sitzen in Ihrer Lieblingskneipe, da kommt ein kleines Menschlein geradewegs auf Sie zu und fängt auch gleich an zu erzählen. Es habe da kürzlich auf seinem Atari in einem C-Programm - oder war es Modula? - einen Dialog konzipiert, der, einmal auf den Bildschirm gebracht, das Publikum durch Intonieren der Händelschen Feuerwerksmusik aufs Trefflichste unterhalten. Sodann mit schnödem Mausklick dazu aufgefordert, habe sein Dialog den Bildschirm in allen Regenbogenfarben schillern und den Umstehenden die Ahs und Ohs aus offnen Mündern nur so kullern lassen, daß es eine Pracht gewesen. Ein weitrer Mausklick und schon erglänzte der Bildschirm von lieblichster Prosa in schönsten Lettern dargebracht...

Noch lange schwärmte das Menschlein von seinem Dialog, und Sie hörten enerviert und ungläubig zu. Vielleicht bestellten Sie dem Kauz auch ein Bier und wechselten die Kneipe. Ein Wort jedoch, ein Zauberwort, vom Kauz beiläufig hingeworfen, blieb Ihnen im Gedächtnis: *G_ProgDef*!

Vielleicht haben Sie auch schon einmal einen Artikel über Submenüs[3] gelesen und auch hier das Zauberwort als Sesam-Öff-

ne-Dich gefunden. *G_ProgDef* (manchmal auch *G_UserDef*) ist ein Objekttyp mit ungeahnten Möglichkeiten. Ein Objekt dieses Typs besitzt nämlich seine eigene Zeichenroutine. Diese Zeichenroutine schreiben Sie sich selbst. Somit hat unser Kauz vielleicht gar nicht übertrieben: Wer verbietet es denn, in einer Routine die Partitur der Feuerwerksmusik über den Soundchip abzuspielen?

Einen Nachteil hat die Sache allerdings: Die Zeichenroutine wird von der Dialogverwaltung des AES aufgerufen. Deshalb muß das Objekt auch über einen Zeiger auf die Adresse seiner eigenen Zeichenroutine verfügen. Wenn Sie einmal den Objekttyp in der Datei GEMTYPE.PAS betrachten, finden Sie darin eine Komponente *ob_spec* vom Typ *Spec_Info*. Diese Komponente besteht, sofern das Objekt vom Typ *G_ProgDef* ist, aus einem Zeiger (ptr: Long_Integer). Dieser Zeiger, so lehrt uns die einschlägige Literatur[4], muß auf eine Struktur vom Typ *Usr_Blk* (Userblock) verweisen. In dieser Struktur (ein Record) finden wir dann eine Komponente *ub_code: Long_Integer* und eine Komponente *ub_parm* vom selben Typ. *Ub_code* aber muß die Adresse (!) der Zeichenfunktion beinhalten.

In ST-Pascal gibt es dummerweise keine legale Möglichkeit, an diese Adresse zu kommen. Auch die Zusatzbibliothek PASTRIX stellt zwar Adreßfunktionen für Variablen zur Verfügung, Prozeduren und Funktionen bleiben aber ausgespart. Was ist zu tun? Wir müßten uns eine Funktion schreiben, die gerade die Adresse einer Funktion oder einer Prozedur ermittelt, genau wie das die PASTRIX-Funktionen mit Variablen machen.

Wie schon angekündigt, gibt es dafür einen (illegalen) Trick. Pascal-Puristen sollten jetzt vielleicht schon einmal Kreuz und Rosenkranz bereithalten und sich, Verzeihung heischend, gen Zürich wenden, wo der Guru der Programmiersicherheit und Verkünder unsrer Lieblingssprache Hof hält.

Um diesen Trick zu verstehen, müssen wir uns erst einmal einiger Kleinigkeiten über modulares Programmieren unter ST-Pascal erinnern. Zu Anfang stellt sich die Frage: Wie funktionieren eigentlich diese PASTRIX-Funktionen? Wie sie genau funktionieren, darüber mag ich nur spekulieren, ich habe sie nie disassembliert. Das ist auch nicht nötig, wenn einem die allgemeine Vorgehensweise klar ist.

Grundlagen

In ST-Pascal gibt es, wie wir jetzt alle wissen, drei Arten von Parametern: 1) die Werteparameter, 2) die Variablenparameter und 3) die Parametrischen Prozeduren und Funktionen. Wird ein Unterprogramm aufgerufen, so legt das aufrufende Programm für die erste Parameterart den Inhalt des Parameters auf den Stack. Für einen Parameter vom Typ Integer sind das immer zwei Bytes, für einen Parameter vom Typ String 81 Bytes. Das aufgerufene Unterprogramm nimmt dann die Menge von Bytes vom Stack, die dem formalen Typ des Parameters entspricht. Durch die strikte Typprüfung bei der Übersetzung kann es dabei nie zu irgendwelchen Unstimmigkeiten kommen.

Die zweite Parameterart (Var) wird immer nur durch einen Zeiger auf die eigentliche Variable repräsentiert. Dieser Zeiger ist, wie alle Zeigertypen, beim Atari immer vier Bytes lang, entspräche also eigentlich dem Typ *Long_Integer*. Dem wird in Pascal Rechnung getragen durch die Möglichkeit, einer Variablen vom Typ *Long_Integer* durch die Ordinalfunktion Ord den „Wert" einer Adresse zu übergeben.

Die dritte Parameterart (Procedure/Function) wird intern genauso gehandhabt wie die zweite: Die Adresse der Funktion oder Prozedur, die übergeben werden soll, wird einfach auf den Stack gebracht und vom aufgerufenen Unterprogramm wieder heruntergeholt. Bringt uns das weiter?

Schlauer Compiler, dummer Linker

Pascal besticht durch seine strikte Überprüfung der Typkompatibilität von formalen und aktuellen Parametern während der Übersetzung. Außerdem stößt jeder Versuch, innerhalb eines Unterprogrammes einen VARiablen-Parameter als Zeiger zu behandeln, auf völliges Unverständnis des Compilers. Diese ganze Prüferei auf Typgleichheit gerät aber in Vergessenheit, kommt der Linker zum Zuge. Dessen Arbeit ist es nämlich nicht, irgendwelche Parametertypen abzuprüfen, sondern nur in der Objektdatei des Hauptprogrammes Unterprogrammnamen zu finden und diese in den ihm angegebenen Objektmodulen und Bibliotheken zu suchen. Der Deklaration des Prozedurkopfes mit der Direktive *external* im Hauptprogramm kommt dabei die Aufgabe zu, sicherzustellen, daß dem Unterprogramm nur die Parameter übergeben werden, die es auch verarbeiten kann. Das gibt uns aber die gewünschte Freiheit: Die Parameterliste des Prozedurkopfes in einem Programm-Modul und diejenige bei der External-Deklaration im Quellcode des Hauptprogrammes können sich unter-

scheiden! Wichtig ist nur, daß die External-Deklaration das Hauptprogramm veranlaßt, bei einem Aufruf des Unterprogrammes exakt dieselbe Menge an Bytes auf den Stack zu bringen, die von dem Unterprogramm dann auch wieder vom Stack geholt wird. Ansonsten würde es bei der Ausführung des Programmes irgendwann einmal Bomben hageln.

Das ist dann auch der illegale Trick, von dem ich sprach: Der Kopf der Funktion *Adr_Proc* in Listing 3 unterscheidet sich von dem derselben Funktion in Listing 4 gerade durch den Typ des Parameters. Da wir aber wissen, daß der Funktion vom Hauptprogramm gerade die Adresse der Prozedur *f* übergeben wird und daß Adressen intern gleich dem Typ *Long_Integer* sind, können wir guten Gewissens diesen Unterschied zu unserem Vorteil einsetzen. So bekommen wir dann **jede** Adresse, je nachdem, wie wir den Parameter in der External-Deklaration wählen. Illegal ist der Trick deshalb, weil die Verschiedenartigkeit der Funktions- und Prozedurköpfe eigentlich nicht Sinn der Sache ist und bei unsachgemäßem Einsatz zu herrlichen Konfusionen Anlaß gibt.

Generelles Vorgehen: Das Programm-Modul aus Listing 3 mit der Direktive *{$M+}* übersetzen und das entstandene Objektmodul als zusätzliche Linkdatei dem Linker angeben. Die Funktion *Adr_Proc* im Hauptprogramm als extern deklarieren und den Parameter je nach Anwendung als VAR-, PROCEDURE- oder FUNCTION-Parameter definieren. Auch der Ergebnistyp der Funktion darf ein beliebiger Zeigertyp oder *Long_Integer* sein. Hier sind Ihnen keine Grenzen gesetzt.

Der Rest ist einfach!?!

Nun, da klar ist, wie man sich Adressen von eigenen Unterprogrammen besorgt, sollte der Verwendung wildester *G_ProgDef*-Objekte nichts mehr im Wege stehen: Zeichenroutine geschrieben, Adresse mit *Adr_Proc* besorgt, in das *UsrBlk*-Record eingehängt - und los geht's... denkste! Die Programmierer von CCD haben uns nämlich die Arbeit abgenommen, mit den Standard-VDI-Routinen umgehen zu müssen, und uns Zeichenprozeduren bereitgestellt, die viel einfacher und sicherer zu handhaben sind. Die können wir aber hier nicht gebrauchen.

Der Grund: Die PASGEM-Routinen verwenden intern eine Routine, die sich mit dem AES überhaupt nicht verträgt. Da unsere Objektroutine aber vom AES bei der Dialogverwaltung (*Do_Dialog* etc.) aufgerufen wird, können wir in ihr auch keine PASGEM-Zeichenroutinen verwenden. Jeder Versuch, eine Standard-PASGEM-Routine innerhalb einer *G_ProgDef*- Zeichenroutine zu benutzen, führt auf einen STACK-OVERFLOW. Zu erklären, warum das so ist, würde den Rahmen dieses Artikels sprengen. Vielleicht werde ich zu einem anderen Anlaß einmal auf diese Problematik zurückkommen.

Wir müssen also auf die originalen VDI-Routinen zurückgreifen, indem wir sie selbst definieren. Die Prozedur *VDI_Call* hilft uns dabei, da sie diese interne Routine nicht verwendet. Merke: *G_ProgDef*-Objekte vertragen keine Standard-PASGEM-Routinen. Außerdem sollten Sie die Debug-Option des Compilers ausschalten, da diese vom AES aus ähnlichen Gründen nicht vertragen wird. Im Beispielprogramm ist das geschehen.

Zum guten Schluß

Nach so viel geballtem Grundlagenwissen will ich Sie dann doch noch mit einem kleinen Demonstrationsdialog belohnen. Das Programm aus Listing 4 baut eine Dialogbox mit zwei Objekten auf: einem Exit-Objekt *DialOk* mit der Inschrift *OK*, um den Dialog zu beenden (ich weiß: sehr sinnig) und dem Objekt der Begierde (oder besser: vom Typ *G_ProgDef*) namens *Special*. Zu diesem Objekt gehört die Prozedur *DrawSpecial*.

Diese Prozedur muß immer einen Parameter vom Typ *ParmBlk* verarbeiten können. Durch seine Komponenten bekommt die Zeichenroutine alle Informationen, die sie benötigt, um ihr Objekt in allen möglichen Zuständen auf den Bildschirm oder sonstwohin zu bringen. Der Parameterblocktyp ist zwar in der Datei GEMTYPE.PAS deklariert, allerdings unsinnigerweise mit der Komponente *pb_tree* vom Typ *long_integer*. Da hier aber der Zeichenroutine der Zeiger (sic!) auf den Dialog (oder das Menü etc.) zugänglich gemacht wird, in dem sich das Objekt befindet, sollte diese Komponente vom Typ *Dialog_Ptr* sein. Aus diesem Grund habe ich, auch wenn ich die Komponente im Beispielprogramm nicht verwende, den Typ neu definiert. Die einzelnen Komponenten des Parameterblocks sind: Der Zeiger auf den kompletten Objektbaum (*PtoDial*), der Index des angesprochenen Objektes (*ObjektNr*), der Objektstatus, der beim letzten Aufruf vorlag (*PrevState*), und der aktuelle Status (*CurrState*). Werden diese beiden Komponenten miteinander verglichen, und unterscheiden sie sich in dem Bit, das anzeigt, ob das Objekt selektiert ist, dann wurde zwischendurch dieses Objekt auf jeden Fall angeklickt. Unterscheiden sich die beiden Komponenten hier nicht, verlangt das AES nur ein einfaches Neuzeichnen des Objektes. Die nächsten vier Komponenten zeigen die Lage und die Größe des Objektes auf dem Bildschirm an. Die Koordinaten *PbX* und *PbY* sind nicht zu verwechseln mit den relativen Koordinaten *ob_x* und *ob_y* im Objekt-Record. Die vier Komponenten *XClip* bis *HClip* stellen den Clipping-Bereich dar, der das Zeichnen des Objektes begrenzen soll. Die Beachtung dieses Bereiches ist notwendig, da durchaus der Fall eintreten kann, daß das Objekt von einem anderen Gegenstand auf dem Schreibtisch teilweise verdeckt wird. Sie sollten also immer einen Clipping-Bereich einstellen, bevor Sie das Objekt zeichnen lassen. Die letzte Komponente ist einfach die Kopie des optionalen Parameters aus dem *UsrBlk*-Record.

Die Arbeitsweise der Prozedur *DrawSpecial* ist relativ banal. Sie zeichnet an die durch *PbX* bis *PbH* angegebene Stelle ein Rechteck mit abgerundeten Ecken und einem Füllmuster, das den Mustern 2 bis 25 der PASGEM-Prozedur *Paint_Style* entspricht. Diese PASGEM-Prozedur vereinigt übrigens gleich zwei VDI-Routinen in sich, aber das ist eine andere Geschichte... Wird das Objekt angeklickt, zählt *DrawSpecial* einfach die globale Variable *SpecialStyle* hoch, um das nächste Füllmuster auszuwählen. „Angeklickt" bedeutet hier, daß sich die beiden Parameterblock-Komponenten *CurrState* und *PrevState* im Bit *Selected* unterscheiden (klar). Als kleinen Gag lassen wir es bei jedem Klick auf das Objekt auch noch einmal klingeln. Vermittels *vsf_style* wählt *SpecialDraw* ein Füllmuster aus, und mit *v_rfbox* zeichnet sie das Rechteck entsprechend den Koordinaten des Parameterblocks. Besteht die Möglichkeit, daß das Objekt teilweise verdeckt ist, muß vorher natürlich noch ein Clip-Bereich eingestellt werden. Bei einem Dialog, der nur mit *Do_Dialog* verwaltet wird, entfällt diese Möglichkeit allerdings. Auch hier gilt: Den Clip-Bereich niemals mit der PASGEM-Prozedur *Set_Clip* einstellen, sondern die entsprechende VDI-Prozedur verwenden.

Der Rest des Programmes ist ebenfalls einfach: Zuerst werden mit *MakeDialog* die Einstellungen zum Zeichnen vorgenommen und der Dialog definiert, wobei *SetUsrBlk* ein *UsrBlk*-Record auf dem Heap erzeugt, die Adresse von *DrawSpecial* einträgt und den Parameter mit 0 vorbelegt. *SetOb_spec* trägt einfach den Zeiger auf das zuvor erzeugte *UsrBlk*-Record in die *ob_spec*-Komponente des Special-Objektes ein. Nachdem der Dialog zentriert und vermittels der AES-Funktion *GrafHandle* der globalen Variablen *VdiHandle*

die Kennung der Workstation zugewiesen wurde, kann der Dialog mit *Do_Dialog* geführt werden. Wird der OK-Knopf angeklickt, beendet das Programm den Dialog, nimmt ihn aus dem Speicher, gibt den Heap-Bereich für den User-Block des Special-Objektes wieder frei und verabschiedet sich von GEM.

Damit wäre ich am Ende angelangt und hoffe, Ihnen ein paar Anregungen zum Umgang mit Parametrischen Prozeduren und mit selbstdefinierten Objekten gegeben zu haben.

Rolf Darr

Literatur:

[1] Goscinny, R und Uderzo, A. „Der Seher". S. 28.

[2] Wirth, Niklaus. „Systematisches Programmieren: Eine Einführung." 3., durchges. Aufl. Stuttgart: Teubner, 1978. (Teubner Studienbücher Informatik; 17). Siehe Abschnitt 12.4 „Parametrische Prozeduren und Funktionen".

[3] Hax, Uwe: „Und es geht doch: Submenüs unter GEM". ST-Computer, 12/89 und 1/ 90

[4] Jankowski, H. D.; Reschke, J. und Rabich, D. „Atari Profibuch". Düsseldorf et al.: Sybex,


```
  1: {------------------------------------------------
  2: Anwendung Parametrischer Prozeduren/Funktionen
  3: zur Automatisierung von Fenster-Redraws bei
  4: Verwendung einer allgemeinen Event-Routine.
  5:
  6: Modul mit Funktion AutoRedraw_Event
  7:
  8: Rolf Darr       (c) 1991 MAXON Computer
  9:
 10: entwickelt mit und für ST-Pascal+ V.2.08 (CCD)
 11: ------------------------------------------------}
 12:
 13: {$I-,C-,D-,P-,T-}   {kein Debugging/Prüfungen}
 14: {$M+} {Modul}
 15: PROGRAM Modul_AutoRedraw_Event;
 16:
 17: CONST
 18:   {$I gemconst.pas}
 19:
 20:
 21: TYPE
 22:   {$I gemtype.pas}
 23:
 24:
 25: {--- die folgenden Subroutinen aus PASGEM ---}
 26: FUNCTION Get_Event( emask, bmask, bstate,
                        n_clicks : short_integer ;
 27:                    ticks : long_integer ;
 28:                    m1_flag : boolean ;
                        m1x, m1y, m1w, m1h :
                           short_integer ;
 29:                    m2_flag : boolean ;
                        m2x, m2y, m2w, m2h :
                           short_integer ;
 30:                   VAR message : Message_Buffer ;
 31:                   VAR key, brtn, bclick,
                           mx, my, kstate :
                           short_integer
 32:                   ) : short_integer ;
 33:   EXTERNAL ;
 34: PROCEDURE Begin_Update ;
 35:   EXTERNAL ;
 36: PROCEDURE End_Update ;
 37:   EXTERNAL ;
 38: PROCEDURE Hide_Mouse ;
 39:   EXTERNAL ;
 40: PROCEDURE Show_Mouse ;
 41:   EXTERNAL ;
 42: PROCEDURE First_Rect( wind : short_integer ;
       VAR x, y, w, h : short_integer ) ;
 43:   EXTERNAL ;
 44: PROCEDURE Next_Rect( wind : short_integer ;
       VAR x, y, w, h : short_integer ) ;
 45:   EXTERNAL ;
 46: FUNCTION Rect_Intersect( x, y, w, h :
      short_integer ; VAR x1, y1, w1, h1 :
      short_integer )
 47:                         : boolean ;
 48:   EXTERNAL ;
 49:
 50:
 51:
 52: {$E+} {----- die neue Eventfunktion -----}
 53: Function AutoRedraw_Event ( EventMask   ,
 54:                             ButtonMask  ,
 55:                             ButtonState ,
 56:                             NClicks : Integer;
 57:                             Ticks : Long_Integer;
 58:                             r1Flag : Boolean;
 59:                             r1x, r1y, r1w, r1h
                                 : Integer;
 60:                             r2Flag : Boolean;
 61:                             r2x, r2y, r2w, r2h
                                 : Integer;
 62:                             VAR Message
                                 : Message_Buffer;
 63:                             VAR Key,
 64:                                 BState, BCount
                                    : Integer;
 65:                             VAR mx, my : Integer;
 66:                             VAR KbdState
                                 : Integer;
 67:
 68:                             PROCEDURE Draw
                                (WindowHandle,
                                x,y,w,h: Integer)
 69:                            ): Integer;
 70:
 71: Const
 72:   MessageType      = 0;
 73:   MesWindowHandle  = 3;
 74:   MesX             = 4;  {Die Koordianten }
 75:   MesY             = 5;  {des zu          }
 76:   MesW             = 6;  {aktualisierenden}
 77:   MesH             = 7;  {Bereiches       }
 78:   NoEvent          = 0;  {kein Event mehr}
 79:
 80: Var
 81:   TempEvent:  Integer;
 82:   {Koordinaten}
 83:   x,y,w,h:    Integer;
 84:
 85: Begin{AutoRedraw_Event}
 86:   REPEAT  {so lange es nur Redraw-Events gibt}
 87:     TempEvent:= Get_Event (
         EventMask|E_Message, {falls noch
                                nicht enthalten}
 88:     ButtonMask, ButtonState, NClicks,
 89:     Ticks,
 90:     r1Flag, r1x, r1y, r1w, r1h,
 91:     r2Flag, r2x, r2y, r2w, r2h,
 92:     Message,
 93:     Key, BState, BCount,
 94:     mx, my,
 95:     KbdState
 96:                            );
 97:     IF TempEvent&E_Message=E_Message
 98:       THEN  {nur auf Redraw-Meldungen
                reagieren}
 99:         IF Message[MessageType] = WM_Redraw
100:           THEN  {den automatischen Redraw
                     organisieren}
101:             BEGIN
102:             {Rechtecklisten abarbeiten}
103:               Begin_Update;
104:               Hide_Mouse;
105:               First_Rect (Message
                     [MesWindowHandle], x,y,w,h);
106:               WHILE (w<>0) AND (h<>0)
107:                 DO
108:                 BEGIN
109:                   IF Rect_Intersect
                         (Message[MesX],
                          Message[MesY],
                          Message[MesW],
                          Message[MesH],x,y,w,h)
110:                     THEN Draw (Message
                     [MesWindowHandle], x,y,w,h);
111:                   Next_Rect (Message
                       [MesWindowHandle], x,y,w,h)
112:                 END;
113:               Show_Mouse;
```
→

```
114:            End_Update;
115:       {und den verarbeiteten Event ausblenden}
116:            TempEvent:= TempEvent &
                             (~E_Message)
117:         END
118:    UNTIL TempEvent <> NoEvent;
119:    AutoRedraw_Event:= TempEvent
120:  End{AutoRedraw_Event};
121: {$E=}
122:
123: BEGIN  {Module haben kein Hauptprogramm}
124: END.
```

```
  1: {-----------------------------------------------
  2: Anwendung Parametrischer Prozeduren/Funktionen
  3: zur Automatisierung von Fenster-Redraws bei
  4: Verwendung einer allgemeinen Event-Routine.
  5:
  6: Beispiel: Aufbau von Fenstern, Definition einer
  7:           allgemeinen Zeichenroutine für die
  8:           Fensterinhalte und Verwaltung via
  9:           neuer AutoRedraw_Event-Routine.
 10:
 11: Rolf Darr       (c) 1991 MAXON Computer
 12:
 13: entwickelt mit und für ST-Pascal+ V.2.08 (CCD)
 14: -----------------------------------------------}
 15: {$I-,C-}
 16: PROGRAM FensterRedrawTest;
 17:
 18: CONST
 19:   {$I gemconst.pas}
 20:   KeinFenster     =    0;
 21:   MaxFenster      =    4;
 22:
 23: TYPE
 24:   {$I gemtype.pas}
 25:   SemaphorTyp     =     KeinFenster..MaxFenster;
 26:
 27: VAR
 28:   {Globale Verwaltung}
 29:   ApplNr          :     Integer;     {klar!}
 30:
 31:   {Globale Fenstervariablen}
 32:   FensterSemaphor :  SemaphorTyp;
 33:   FensterTitel    :  Window_Title;
 34:
 35:
 36: {$I gemsubs.pas}   {alle Routinen aus PASGEM}
 37:
 38: { Zusätzlich noch die neue Auto-Redraw-Event-
      Funktion}
 39: {===============================================
 40:  DIESEN TEIL IN EINE INCLUDE-DATEI SPEICHERN ODER
 41:  AN DAS ENDE DER DATEI GEMSUBS.PAS HÄNGEN.
 42:  DIE OBJEKTDATEI DES PROGRAMMODULS AUS LISTING 1
 43:  ALS ZUSÄTZLICHE LINKDATEI EINTRAGEN UND DIE
 44:  NEUE FUNKTION IMMER DANN STATT DER ALTEN VER-
 45:  WENDEN, WENN FENSTERINHALTE AUTOMATISCH GEZEICH-
 46:  NET WERDEN SOLLEN.
 47: ===============================================}
 48: FUNCTION AutoRedraw_Event
 49:          ( EventMask, ButtonMask,
 50:            ButtonState, NClicks: Short_Integer;
 51:            Ticks                : Long_Integer;
 52:            r1Flag               : Boolean;
 53:            r1x, r1y, r1w, r1h   : Short_Integer;
 54:            r2Flag               : Boolean;
 55:            r2x, r2y, r2w, r2h   : Short_Integer;
 56:            VAR Message          : Message_Buffer;
 57:            VAR Key,
 58:                BState, BCount   : Short_Integer;
 59:            VAR mx, my           : Short_Integer;
 60:            VAR KbdState         : Short_Integer;
 61:
 62:            PROCEDURE Draw (WindowHandle,
 63:                            x,y,w,h:Short_Integer)
 64:          ):  Short_Integer;
 65:   EXTERNAL;
 66: {===============================================}
 67:
 68: {----- Eine allgemeine Zeichenfunktion für
        Fensterinhalte -----}
 69: Procedure FensterDraw ( Handle, x,y,w,h:
                                   Integer );
 70:   Begin{FensterDraw}
 71:    Paint_Color ( White );
 72:    Line_Color ( Black );
 73:    Draw_Mode ( 1 );
 74:    Set_Clip( x, y, w, h );
 75:    Paint_Rect( x, y, w, h );        { leeren }
 76:    Frame_Rect( x, y, w, h );        { Kasten }
 77:    PLine( x, y, x+w-1, y+h-1 ); { X in Kasten }
 78:    PLine( x+w-1, y, x, y+h-1 );
 79:   End{FensterDraw};
 80:
 81:  {----- Fenster öffnen -----}
 82:  Function FensterAuf (FensterSemaphor:
                     SemaphorTyp ): SemaphorTyp;
 83:   Var
 84:     hand: integer;
 85:   Begin{FensterAuf}
 86:     IF FensterSemaphor < MaxFenster
 87:       THEN
 88:         BEGIN
 89:           FensterSemaphor:=
                        succ (FensterSemaphor );
 90:           hand:= New_Window
               (G_Name|G_Close|G_Move|G_Size,
               FensterTitel, 0,0,0,0);
 91:           Open_Window (hand, 20*FensterSemaphor,
               20*FensterSemaphor, 200,120);
 92:         END;
 93:     FensterAuf:= FensterSemaphor
 94:   End{FensterAuf};
 95:
 96:  {----- Initialisierungen -----}
 97:  Procedure FensterInit (VAR semaphor:SemaphorTyp);
 98:   Begin{FensterInit}
 99:     FensterTitel:=  ' Fenster ';
100:     semaphor:=  KeinFenster;
101:     REPEAT
102:       semaphor:=  FensterAuf ( semaphor )
103:     UNTIL semaphor = MaxFenster
104:   End{FensterInit};
105:
106:  {----- Fenster schliePen -----}
107:  Procedure FensterZu ( hand: Integer );
108:   Begin{FensterZu}
109:     close_window ( hand );
110:     delete_window ( hand )
111:   End{FensterZu};
112:
113:  {----- Die Eventverwaltung -----}
114:  Procedure event_loop;
115:
116:   Const
117:     M_Type      =     0;     {Message Index}
118:   Var
119:     event:                integer;
120:     msg:                  message_buffer;
121:     key,
122:     bcnt,bstate, mx,my,
123:     kbd_state:            integer;
124:
125:  s:string;dummy:integer;
126:
127:   Begin
128:     REPEAT
129:       event:= AutoRedraw_Event
130:               ( E_Keyboard|E_Message,
131:                 0,0,0,              { diverse }
132:                 0,                  { timer }
133:                 false,0,0,0,0,      { mouse(r1) }
134:                 false,0,0,0,0,      { mouse(r2) }
135:                 msg,            { message- buffer }
136:                 key,              { keyboard-code }
137:                 bcnt,bstate, mx,my, { mouse-
                                          states }
138:                 kbd_state, { sondertasten }
139:                 FensterDraw { die Zeichen-
                                 routine}
140:               );
141:
142:       IF event&E_Message = E_Message
143:         THEN
144:           CASE msg[M_Type]
145:           OF
146:             WM_Topped:  Bring_To_Front (msg[3]);
147:             WM_Closed:  BEGIN
148:                           FensterZu (msg[3]);
```

→

```
149:                    FensterSemaphor:=
                          pred(FensterSemaphor)
150:                  END;
151:            WM_Sized,
152:            WM_Moved:   Set_WSize (msg[3],
                        msg[4],msg[5],msg[6],msg[7])
153:          END{ Case };
154:
155:        IF event&E_Keyboard = E_Keyboard
156:          THEN {Tastatur-Verwaltung}
157:            BEGIN
158:              WriteV ( s, '[1][Taste = $',key:4:h,
                        ' ][  OK  ]' );
159:              Dummy:= Do_Alert (s,1)
160:            END;
161:      UNTIL FensterSemaphor = KeinFenster;
162:    End{ event_loop };
163:
164:  {--------------- Hauptprogramm ---------------}
165:  BEGIN
166:    ApplNr:=  Init_Gem;
167:    IF ApplNr>=0
168:      THEN
169:        BEGIN
170:          Init_Mouse;
171:          FensterInit (FensterSemaphor);
172:          event_loop;
173:          exit_gem
174:        END
175:  END{Hauptprogramm}.
```

```
 1:  {-----------------------------------------------
 2:  Anwendung Parametrischer Prozeduren/Funktionen
 3:  zur Erzeugung von Prozeduradressen.
 4:
 5:  Modul mit externer Funktion Adr_Proc
 6:
 7:  Rolf Darr     (c) 1991 MAXON Computer
 8:
 9:  entwickelt mit und für ST-Pascal+ V.2.08 (CCD)
10:
11:  -----------------------------------------------}
12:  {$I-,C-,D-,P-,R-,T-} {kein Debugging/Prüfungen}
13:  {$M+}                {Modul}
14:  PROGRAM Modul_AdrProc;
15:
16:  {$E+}
17:  FUNCTION Adr_Proc ( Parameter: Long_Integer ):
       Long_Integer;
18:    BEGIN{Adr_Proc}
19:      Adr_Proc:=  Parameter {silly one!!!}
20:    END{Adr_Proc};
21:  {$E=}
22:
23:  BEGIN    {Module haben kein Hauptprogramm}
24:  END.
```

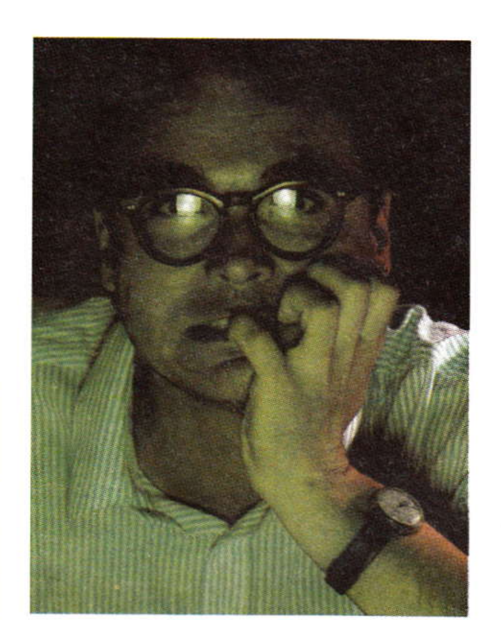
```
 1:  {-----------------------------------------------
 2:  Anwendung Parametrischer Prozeduren/Funktionen
 3:  zur Erzeugung von Prozeduradressen.
 4:
 5:  Beispiel: Einbau einer eigenen Zeichenroutine
 6:            in ein Dialogitem vom Typ G_ProgDef.
 7:
 8:  Rolf Darr    (c) 1991 MAXON Computer
 9:
10:  entwickelt mit und für ST-Pascal+ V.2.08 (CCD)
11:
12:  -----------------------------------------------}
13:  {$I-,C-,D-,P-,R-,T-}   {kein Debugging/Prüfungen}
14:  PROGRAM DialogTest;
15:
16:  CONST
17:    {$I gemconst.pas}
18:
19:
20:  TYPE
21:    {$I gemtype.pas}
22:
23:    ParmBlkPtr=    ^ParmBlk;
24:    ParmBlk    =   RECORD
```
→

```
25:                 PtoDial:        Dialog_Ptr;
                     {Zeiger auf Objektbaum}
26:                 ObjNr:          Integer;
                     {Objektnummer}
27:                 PrevState:      Integer;
                     {vorheriger Status}
28:                 CurrState:      Integer;
                     {neuer Status}
29:                 PbX:            Integer;
                                     {Koordinaten}
30:                 PbY:            Integer;
31:                 PbW:            Integer;
32:                 PbH:            Integer;
33:                 XClip:          Integer;
                               {Begrenzungsrechteck}
34:                 YClip:          Integer;
35:                 WClip:          Integer;
36:                 HClip:          Integer;
37:                 PbParm:         Long_Integer;
                               {Parameter aus USRBLK}
38:               END;
39:
40:     UsrBlkPtr =       ^User_Blk;
41:
42:
43:  VAR
44:     {Globale Verwaltung}
45:     ApplNr     :    Integer;    {klar!}
46:     VdiHandle  :    Integer;    {der Workstation}
47:     {Für das Special-Item:}
48:     SpecialStyle  :  Integer;  {Index Füllmuster}
49:     UbSpecial  :    UsrBlkPtr; {User-Block}
50:
51:     {Variablen für den Dialog}
52:     MyDialog      :   Dialog_Ptr;
53:     {Dialog-Items}
54:     ErgItem,                    {nur als Dummy}
55:     DialOk,                     {Dialog-Ausgang}
56:     Special       :   Integer;  {ProgDef-Item}
57:
58:
59:
60:  {$I gemsubs.pas}    {alle Routinen aus PASGEM}
61:
62:  { zusätzlich benötigen wir noch folgende VDI-
     Routinen innerhalb der Zeicheroutine für das
63:    Special-Item, da wir die entsprechenden PASGEM-
64:    Routinen leider
65:    nicht verwenden können.}
66:
67:  {- v_rfbox : Filled Rounded Rectangle VDI 11/9 -}
68:  PROCEDURE v_rfbox ( Handle: Integer; x1,y1, x2,
                           y2:Integer );
69:    CONST
70:      Opcode       = 11;
71:      SubOpCode    = 9;
72:      AnzIntIn     = 0;
73:      AnzPunkte    = 4;
74:    VAR
75:      Ctrl      : Ctrl_Parms ;
76:      IntIn     : Int_In_Parms ;
77:      IntOut    : Int_Out_Parms ;
78:      PtsIn     : Pts_In_Parms ;
79:      PtsOut    : Pts_Out_Parms ;
80:    BEGIN{v_rfbox}
81:      Ctrl[6]:= Handle;
82:      PtsIn[0]:=  x1;
83:      PtsIn[1]:=  y1;
84:      PtsIn[2]:=  x2;
85:      PtsIn[3]:=  y2;
86:      VDI_Call  ( Opcode,SubOpCode,AnzIntIn,
                  AnzPunkte, Ctrl, IntIn, IntOut,
87:               PtsIn, PtsOut, False )
88:    END{v_rfbox};
89:
90:  {- vsf_style :- Set Fill Style Index VDI 24 -}
91:  PROCEDURE vsf_style ( Handle: Integer;
                     StyleIndex: Integer );
92:    CONST
93:      Opcode       = 24;
94:      SubOpCode    = 0;
95:      AnzIntIn     = 1;
96:      AnzPunkte    = 0;
97:    VAR
98:      Ctrl      : Ctrl_Parms ;
99:      IntIn     : Int_In_Parms ;
100:     IntOut    : Int_Out_Parms ;
101:     PtsIn     : Pts_In_Parms ;
102:     PtsOut    : Pts_Out_Parms ;
103:   BEGIN{vsf_style}
104:     Ctrl[6]:= Handle;
105:     IntIn[0]:=  StyleIndex;
106:     VDI_Call ( Opcode,SubOpCode,AnzIntIn,
                  AnzPunkte,Ctrl, IntIn, IntOut,
107:              PtsIn,PtsOut, False )
108:   END{vsf_style};
109:
110: {sowie die AES-Funktion:}
111:
112: {----- Graf_Handle (AES 77) -----}
113: FUNCTION GrafHandle ( VAR cw, ch,  bw, bh:
                       Integer ): Integer;
114:   VAR
115:     IntIn:  Int_In_Parms;
116:     IntOut: Int_Out_Parms;
117:     AdrIn:  Addr_In_Parms;
118:     AdrOut: Addr_Out_Parms;
119:   BEGIN{GrafHandle}
120:     AES_Call (77, IntIn, IntOut, AdrIn, AdrOut);
121:     GrafHandle:= IntOut[0];
122:     cw:= IntOut[1];
123:     ch:= IntOut[2];
124:     bw:= IntOut[3];
125:     bh:= IntOut[4];
126:   END{GrafHandle};
127:
128:
129: {Die externe Funktion aus dem Modul von Listing 3
130:  hier für unsere Zwecke umdeklariert:}
131: FUNCTION Adr_Proc  ( PROCEDURE f(pb: ParmBlkPtr)
                   ): Long_Integer;
132:   EXTERNAL;
133:
134:
135: {-die Draw-Routine für das SPECIAL-ITEM -}
136:
137: PROCEDURE DrawSpecial   ( pb: ParmBlkPtr );
138:   CONST
139:     LastStyle =   24;
140:   BEGIN{DrawSpecial}
141:     WITH pb^
142:       DO
143:       BEGIN
144:         IF CurrState&Selected<>
                 PrevState&Selected  {angeklickt?}
145:         THEN
146:           BEGIN
147:             {nächstes Füllmuster: wir zählen
                  einfach den Index
148:              von 1 bis 24 im Rundlauf}
149:             SpecialStyle:=  (SpecialStyle+1)
                  MOD LastStyle;
150:             Write (#7); {und klingeln}
151:             {Füllmuster setzen}
152:           END;
153:         vsf_style  ( VdiHandle, SpecialStyle+1 );
154:         {Normalerweise müßte hier noch der Clip-
             Bereich eingestellt
155:          werden, was natürlich auch nicht mit
             der PASGEM-Routine
156:          "Set_Clip" geschehen darf, sondern mit
             der extra definierten
157:          VDI-Routine "Set clipping rectabgle"
             (129). Hier unnötig!}
158:         {Gebilde ganz zeichnen}
159:         v_rfbox (VdiHandle, PbX, PbY, PbX+PbW,
                     Pby+PbH);
160:       END
161:   END{DrawSpecial};
162:
163:
164: {------------ Dialog erstellen -------------}
165:
166: Procedure SetUsrBlk; {trägt die
                          Funktionsadresse und den
                          Wert des optionalen
                          Parameters in die UsrBlk-
168:                      Strukur ein}
169:   Var
170:     FunctionAdress: Long_Integer;
171:   Begin{SetUsrBlk}
172:     New (UbSpecial);
173:     FunctionAdress:=  Adr_Proc  ( DrawSpecial );
174:     WITH UbSpecial^
175:       DO
176:       BEGIN
177:         ub_code:= FunctionAdress;
178:         ub_parm:=  0
179:       END
180:   End{SetUsrBlk};
181:
182: Procedure SetOb_spec ( VAR Obj: Object );
```

→

```
183:                          {hängt die globale
                               UsrBlk-Struktur in
184:                           den Ob_Spec-Zeiger
                               eines Objektes ein}
185:    Begin{SetOb_spec}
186:      Obj.ob_spec.ptr:= Ord( UbSpecial );
187:    End{SetOb_spec};
188:
189:  Function MakeDialog: Dialog_Ptr;
                              {Beispiel-Dialog erstellen}
190:    Const
191:      MaxItems    =    5; {etwas Luft nach hinten}
192:      Xpos        =    0; {wird sowieso zentriert}
193:      Ypos        =    0;
194:      W           =   36; {kann verändert werden}
195:      H           =   12;
196:    Var
197:      Dial:   Dialog_Ptr;
198:      Dummy:  Integer;
199:    Begin{MakeDialog}
200:      {Einstellungen zum Zeichnen. Hier können
           noch die PASGEM-Routinen verwendet werden,
201:       weil wir noch nicht innerhalb
202:       des AES sind!}
203:      Paint_Style ( 2 );  {setzt intern den
                               Fülltyp auf 2 und
204:                           den Musterindex auf 1.
                               Sparte die
205:                           Deklaration von
                               SetFillInteriorIndex
206:                           (VDI 23), daher nötig}
207:      {Die anderen Einstellungen}
208:      Paint_Color ( Black );
209:      Draw_Mode ( 1 {Replace!} );
210:      Line_Style ( 1 );
211:      Paint_Outline ( True );
212:
213:      {Speicherplatz reservieren}
214:      Dial:= New_Dialog (MaxItems,Xpos,Ypos,W,H);
215:
216:      {Die Items eintragen}
217:      DialOk:= Add_Ditem ( Dial, G_BoxText,
            Selectable|Default|Exit_Btn,
218:        (W DIV 2)-5, H-3, 10,2,-2,
219:        D_Color(Black,Black,True, 0, 0)
220:                         );
221:      Set_Dtext ( Dial, DialOk, 'OK', System_Font,
            TE_Center );
222:      Obj_SetState (Dial,DialOk,Shadowed,False);
223:
224:      Special:= Add_Ditem ( Dial, G_ProgDef{!!!},
                                 Selectable,
225:                             2, 1, W-4, H-6, 0, 0);
226:      Obj_SetState (Dial,Special,Normal,False);
227:      SpecialStyle:=  0; {nur als Initialisierung}
228:      {Die Spezialfunktion einbinden...}
229:      SetUsrBlk;
230:      SetOb_spec ( Dial^[Special]);
231:
232:      {Den Dialog in die Mitte bringen}
233:      Center_Dialog ( Dial );
234:
235:      {Globalen Vdi-Handle für Draw-Funktionen
           holen}
236:      VdiHandle:= GrafHandle (Dummy,Dummy,Dummy,
           Dummy);
237:
238:      {und fertig: den Zeiger retournieren}
239:      MakeDialog:=  Dial
240:    End{MakeDialog};
241:
242:  {--------------- Hauptprogramm ---------------}
243:  BEGIN
244:    ApplNr:=  Init_Gem;
245:    IF ApplNr>=0
246:      THEN
247:        BEGIN
248:          Init_Mouse;
249:          MyDialog   :=  MakeDialog;
250:          ErgItem    :=  Do_Dialog (MyDialog,0);
251:          End_Dialog (MyDialog);
252:          Delete_Dialog (MyDialog);
253:          Dispose (UbSpecial);   {sollte noch
                                      freigegeben werden}
254:          exit_gem
255:        END
256:  END{Hauptprogramm}.
```

SCREEN WATCH



Direkter Bildschirm-zugriff? Nein danke!

Die gegenüber dem ST neuen Bildschirmauflösungen des TT haben ein ganz spezielles Problem aufgezeigt: Einige Programme greifen direkt auf den Bildschirmspeicher zu und umgehen somit das GEM. Dies kann zu Fehlern und Inkompatibilitäten führen. Solche Übeltäter können auf dem TT entlarvt werden.

Aber nicht nur TT-Anwender, sondern auch Besitzer eines ST haben mit Programmen zu kämpfen, die die Nutzung neuer Auflösungen durch direkte Zugriffe auf den Bildschirmspeicher verbauen. Schließlich gibt es für den ST inzwischen diverse Hardware-Lösungen, die es erlauben, auf dem SM124 eine höhere Auflösung als 640x400 Punkte zu nutzen. Ärgerlich, wenn man feststellen muß, das ein Programm, für das eine höhere Auflösung von Vorteil wäre, nicht mit dem größeren Bildschirm zurechtkommt. Gerade bei Textverarbeitungen profitiert man von jeder Textzeile oder -spalte, die zusätzlich auf dem Bildschirm untergebracht werden kann.

Unsauber programmiert

Anders kann man Programme, die durch einen direkten Zugriff auf den Bildschirmspeicher die Nutzung neuer Auflösungen verhindern, nicht bezeichnen. Bereits seit einiger Zeit (spätestens seit der Ankündigung des TT) ist klar, daß Text- und Grafikausgaben möglichst nicht unter Umgehung des GEM erfolgen sollten. Einige Programme reagieren zwar auch ohne GEM-Nutzung sehr flexibel auf unterschiedliche Auflösungen, aber spätestens dann, wenn ein Bildschirm vorliegt, der nicht auf Bitmap-Basis arbeitet (z.B. ein grafikfähiges Terminal), führen eigene Ausgaberoutinen nicht mehr zum Ziel. Zwar sind solche Bildschirme für ST oder TT noch nicht aktuell, aber das kann sich ja noch ändern.

Sicherlich muß man bei konsequentem Einsatz des GEM einen gewissen Geschwindigkeitsverlust bei der Grafikdarstellung in Kauf nehmen, aber werden AES und VDI optimal programmiert, sind durchaus hohe Geschwindigkeiten möglich. Gerade bei den Rechnern der neueren Generation, also beim Mega STE und TT, werden auch bei ausschließlicher Nutzung des GEM gute Zeiten bei der Bildschirmausgabe erreicht.

Nicht todernst

Zwar kann man direkte Zugriffe auf den Bildschirmspeicher nicht sinnvoll verhindern, aber Besitzer des TT haben immmerhin die Möglichkeit, Programme, die das GEM durch solche Methoden umgehen zu entlarven. Wenn auch diese Anwendung eine nicht ganz ernste ist, gibt sie uns doch eine gute Gelegenheit, neue Seiten der PMMU des 68030 kennenzulernen: Deskriptoren im Langformat und den Zugriffsschutz für einzelne Speicherseiten.

Lange Deskriptoren

In den letzten Ausgaben der ST-Computer haben uns in Verbindung mit der MMU ausschließlich Deskriptoren mit einer Länge von 32 Bits beschäftigt. Nun gibt es neben diesem Kurzformat jedoch auch noch ein Deskriptor-Langformat, das gleich 64 Bits in Anspruch nimmt. Eines der neu hinzugekommenen Bits erlaubt es, einzelne Speicherseiten gezielt gegen einen Zugriff aus dem User-Modus zu schützen. Es handelt sich um das *S-Bit* (Supervisor Only).

Der Aufbau dieser für uns neuen Deskriptoren wird in Bild 1 dem bereits bekannten kurzen Format gegenübergestellt. Mit welchem Deskriptor-Typ die MMU zu rechnen hat, wird im *CRP*-Register (CPU Root Pointer) anhand des *DT*-Feldes (Descriptor Type) definiert.

Bildschirmschutz

Wie kann man einen Schreibschutz für den Bildschirmspeicher nun konkret verwirklichen? Der naheliegende Gedanke, den Bildschirm einfach in einem schreibgeschützten Speicherbereich anzusiedeln, läßt sich zwar (bereits mit kurzen Deskriptoren) verwirklichen, ist allerdings wenig günstig. Fehler beim Schreiben auf den Bildschirm würden dann nämlich auch vom Betriebssystem verursacht, da ja das Ansprechen des Bildschirmspeichers unter keinen Umständen erlaubt wäre. Ausgaben, die über das GEM gemacht werden, sollten jedoch nicht irgendwelchen Einschränkungen unterliegen.

Da ist es schon besser, den Bildschirm nicht global zu schützen, sondern lediglich Zugriffe aus dem User-Modus abzufangen und diese durch einen Signalton anzuzeigen. Dabei setze ich voraus, daß

```
+---------------------------------------------+
|        Tabellen-Adresse, Bits 31-16         |
| Tabellen-Adresse, Bits 15-4  | U |WP| 1 | 0 |   Tabellen-Deskriptor, Kurzformat
+---------------------------------------------+

+---------------------------------------------------------+
|LU|                   Index-Limit                        |
| 1| 1| 1| 1| 1| 1| 0| S| 0| 0| 0| 0| U|WP| 1| 1|           |
|        Tabellen-Adresse, Bits 31-16                     |   Tabellen-Deskriptor, Langformat
| Tabellen-Adresse, Bits 15-4        | Unbenutzt          |
+---------------------------------------------------------+

+---------------------------------------------+
|         Seiten-Adresse, Bits 31-16          |
|     Bits 15-4      | 0|CI| 0| M| U|WP| 0| 1 |   Seiten-Deskriptor, Kurzformat
+---------------------------------------------+

+---------------------------------------------------------+
|                      Unbenutzt                          |
| 1| 1| 1| 1| 1| 1| 0| S| 0|CI| 0| 0| U|WP| 0| 1|           |
|         Seiten-Adresse, Bits 31-16                      |   Seiten-Deskriptor, Langformat
|     Bits 15-8      |          Unbenutzt                 |
+---------------------------------------------------------+
```

Bild 1: Deskriptoren

```
+-------------------------------------------------------------+
|LU|                    Index-Limit                           |
| 0| 0| 0| 0| 0| 0| 0| 0| 0| 0| 0| 0| 0| 0|    DT             |
|          Tabellen-Adresse, Bits 31-16                       |
| Tabellen-Adresse, Bits 15-4         | Unbenutzt             |
+-------------------------------------------------------------+
CPU Root Pointer Register

|RR|//|IF|DF|RM|HB|BY|RW|//|//|//|//|//|FC2-FC0|
Special Status Register
```

Bild 2: CRP und SSR

ein Anwenderprogramm in der Regel im User-Modus läuft. Ist dies nicht der Fall, erfolgen Zugriffe auf den Bildschirmspeicher also im Supervisor-Modus, läßt sich natürlich durch den hier in Betracht gezogenen Schutzmechanismus nichts erreichen.

The same procedure ...

Wie bereits in [1] erläutert, erlaubt es der RTE-Befehl, eine Operation, die zu einer Exception geführt hat, nochmals auszuführen. Für SCREENWATCH genügt dies alleine aber nicht, denn es würde natürlich sofort wieder ein Busfehler resultieren. Anders sieht es jedoch aus, wenn der nächste Befehl, also nur der Zugriff auf den Bildschirm, ausnahmsweise im Supervisor-Modus erfolgen könnte. Der 68030 bietet in der Tat die Möglichkeit, die Zugriffsbedingungen für die Wiederholung des Befehls, der zum Busfehler führte, zu ändern. Alle Daten, die hierzu benötigt werden, befinden sich während der Exception auf dem Interrupt-Stack [2]. An Offset 10 relativ zum Stackpointer befindet sich das Special Status-Register, das Aussagen über den internen Zustand des Prozessors zum Zeitpunkt des Fehlers macht.

Spezialitäten

Um solche handelt es sich bei den Bits des Special Status-Registers (Bild 2), die Aussagen über einen Buszyklus machen, bei dem ein Busfehler auftrat:

RR (Re-Run): Ist dieses Bit gesetzt, was der Default-Zustand ist, wird bei der Ausführung von RTE der letzte Buszyklus wiederholt.
IF (Instruction Fetch): Der Fehler trat beim Holen des nächsten Befehls auf.
DF (Data Fetch): Der Busfehler wurde beim Holen eines Datenwortes erzeugt.
RM (Read/Modify/Write): Der Fehler entstand während eines Read/Modify/Write-Zyklus'.
HB (High Byte/Low Byte): Bei gesetztem Bit trat der Busfehler auf, als auf das High-Byte eines Wortes zugegriffen wurde.
BY (Byte/Word): Ist dieses Bit 1, wurde beim Auftreten des Fehlers auf ein Byte zugegriffen, andernfalls handelte es sich um einen Wortzugriff.
RW (Read/Write): Der Fehler wurde während eines Lesezugriffs hervorgerufen, wenn das RW-Bit gesetzt ist.
FC2-FC0 (Function Code Bits): Diese Bits machen eine nähere Aussage über die Art des Zugriffs (s.u.)

Alle weiteren Bits des Special Status-Registers sind nicht belegt.

User oder Supervisor?

Es ist programmgesteuert möglich, die Bedingungen, unter denen der nächste Daten- oder Adreßzugriff erfolgen soll, nahezu beliebig zu verändern. Da sich alle Angaben über den internen Zustand des Prozessors auf dem Stack befinden, sind weitreichende Manipulationen denkbar (und erlaubt!). In unserem Fall sind lediglich die Function Code Bits *FC2-FC0* von Bedeutung, die Aussagen über Art und Adreßraum des Zugriffs machen:

FC2-0	Adreßraum
000	reserviert
001	User-Daten
010	User-Programm
011	reserviert
100	reserviert
101	Supervisor-Daten
110	Supervisor-Programm
111	CPU-Adreßraum

Es läßt sich leicht erkennen, daß *FC2* bestimmt, ob ein Zugriff unter den Bedingungen des User- oder Supervisor-Modus' stattfindet. Wird das entsprechende Bit im Statuswort gesetzt, gelten für die Wiederholung des Befehls durch RTE die Zugriffsrechte des Supervisor-Modus'. SCREENWATCH wird also nur für den nächsten Befehl keine Busfehler-Meldung erhalten. Und genau das wollten wir ja erreichen.

Langgemacht

Da ein Zugriffsschutz speziell für den User-Modus nur mit langen Seitendeskriptoren zu realisieren ist, besteht eine der Hauptaufgaben von SCREENWATCH darin, die kurzen Deskriptoren der Standardtabelle in das Langformat umzuwandeln.

SCREENWATCH versucht bei der Konvertierung so vorzugehen, daß bereits geänderte Deskriptoren ihre Werte behalten. So wird eine möglichst hohe Kompatibilität zu anderen Programmen erzielt, falls diese ebenfalls Gebrauch von der MMU machen. Die in den kurzen Deskriptoren enthaltenen Adressen weiterer Deskriptoren oder Speicherseiten werden vom Programm von den Deskriptor-Flags isoliert und anschließend zu langen Deskriptoren zusammengesetzt.

Nicht erwischt

SCREENWATCH ist in der Lage, die meisten unsauberen Zugriffe auf den Bildschirm zu erkennen. Ausnahmen stellen, wie bereits angesprochen, Programme dar, die aus dem Supervisor-Modus heraus auf den Bildschirm zugreifen. In solchen Fällen ist der Schreibschutz auf den Bildschirmspeicher erlaubt, da sonst das GEM ja keine Möglichkeit hätte, Ausgaben ohne Warnmeldung vorzunehmen.

Ebenfalls nicht erkannt werden Zugriffe, die vorgenommen werden, nachdem ein Programm den Bildschirmspeicher an eine neue RAM-Adresse gelegt hat. SCREENWATCH kann schließlich nur den Speicherbereich schützen, in den das Programm während der Initialisierung den neuen Bildschirm legt. Wird diese Adresse durch nachfolgende Programme geändert, wird der Schutzmechanismus deaktiviert.

MMU-Schlamassel

Bei der Verwendung von SCREENWATCH ist zu beachten, daß möglichst keine anderen Programme installiert werden sollten, die ebenfalls Gebrauch von der MMU machen. Ganz allgemein kann man jedoch davon ausgehen, daß es Schwierigkeiten gibt, wenn mehr als ein Programm die MMU für sich zu beanspruchen versucht. Ferner darf kein Programm vor SCREENWATCH gestartet worden sein, das eine eigene Busfehler-Routine einrichtet, die nicht den in [1] aufgestellten Kriterien genügt.

SCREENWATCH geht davon aus, daß die Deskriptor-Tabellen beim Programmstart so vorliegen, wie es beim Atari TT der Normalzustand ist. Leider liegt zu diesen Tabellen noch keine offizielle Dokumentation von Atari vor. Dennoch machen bereits einige Programme von der Tatsache Gebrauch, daß TOS die Deskriptoren im Speicherbereich von $0700 bis $0800 anlegt. Bleibt zu hoffen, daß demnächst Informationen erhältlich sind, die eine legale Nutzung dieser Tabellen zulassen. Leider lassen solche Unterlagen gewöhnlich lange auf sich warten ...

US

Literatur:

[1] „Virtuelle Speicherverwaltung - Eine Fallstudie", ST-Computer 6/91

[2] Steve Williams, „68030 Assembly Language Reference", Addison-Wesley Publishing Company Inc., ISBN 0-201-08876-2


```
 1:  *********************************
 2:  *                               *
 3:  * SCREENWATCH V1.0              *
 4:  *                               *
 5:  * Überwachung des Bildschirms   *
 6:  *                               *
 7:  * per MMU                       *
 8:  *                               *
 9:  * by Uwe Seimet                 *
10:  * (c) 1991 MAXON Computer       *
11:  *********************************
12:
13:
14:  GEMDOS   = 1
15:  CCONWS   = 9
16:  SUPER    = 32
17:  PTERMRES= 49
18:  MXALLOC = 68
19:  MSHRINK = 74
20:
21:
22:  BCONOUT = 3
23:  BIOS     = 13
24:
25:
26:  XBIOS    = 14
27:  SETSCREEN=5
28:
29:
30:  _v_bas_ad= $44e
31:  _p_cookies= $5a0
32:
33:
34:  magic    = "SCRW"                 ;für XBRA
35:
36:
37:          text
38:
39:          move.l 4(sp),a0           ;Pointer auf
40:                                    ;Basepage
41:          move.l 12(a0),a6
42:          add.l 28(a0),a6
43:          lea $100(a6),a6
44:          pea (a6)
45:          pea (a0)
46:          clr -(sp)
47:          move #MSHRINK,-(sp)       ;Restspeicher
48:          trap #GEMDOS              ;freigeben
49:          lea 12(sp),sp
50:          tst.l d0
51:          bne quit                  ;Fehler-
52:
53:          clr.l -(sp)
54:          move #SUPER,-(sp)         ;Supervisor-
55:          trap #GEMDOS              ;Modus
56:          addq.l #6,sp
57:          move.l d0,d7              ;SSP merken
58:
59:          move.l $08,a0
60:          cmp.l #magic,-8(a0)       ;bereits
61:          beq exit                  ;installiert-
62:
63:          lea sterr(pc),a5
64:          move.l _p_cookies,d0
65:          beq error                 ;kein cookie jar
66:          move.l d0,a0
67:  nomch:  movem.l (a0)+,d0-d1
68:          tst.l d0                  ;Ende des
69:          beq error                 ;cookie jar-
70:          cmp.l #"_MCH",d0          ;Computertyp?
71:          bne nomch                 ;nein-
72:          swap d1
73:          subq.l #2,d1              ;TT?
74:          bne error                 ;nein-
75:
76:          lea memerr(pc),a5
77:          clr -(sp)                 ;nur ST-RAM
78:          pea 5*32768+32767         ;Platz für 5
              Pages
79:          move #MXALLOC,-(sp)       ;anfordern
80:          trap #GEMDOS
81:          addq.l #8,sp
82:          add.l #32767,d0
83:          and #$8000,d0             ;Bildschirmstart
84:                                    ;auf Page-Anfang
85:          move.l d0,d5              ;neue Bildschirm-
86:                                    ;Adresse
87:          beq error              ;kein Speicher-   →
```

```
88:
89:           move.l _v_bas_ad,a0
90:           move.l d5,a1
91:           move #38399,d0
92:  scrcpy:  move.l (a0)+,(a1)+      ;Bildschirm
93:           dbra d0,scrcpy           ;kopieren
94:
95:           move.l #table+15,d6
               ;Deskriptortabelle
96:           and #$fff0,d6           ;auf 16-Byte-
                                       Grenze
97:
98:           pmove crp,crpsav         ;alter CRP
99:           move.l crpsav+4,a0       ;Pointer auf
100:                                   ;alte Tabelle
101:          move.l d6,crpreg+4       ;für neuen CRP
102:
103:          moveq #63,d0             ;64 Deskriptoren
104:          move.l d6,a1             ;neue Tabelle
105:          move.l crpsav+4,a2       ;zeigt auf
106:          lea 256(a2),a2           ;Tabellenende
107: copy:    move.l (a0)+,d1          ;Kurzformat
108:          move.l d1,d2
109:          and.l #$ff,d1            ;Adresse
                                       ausblenden
110:          btst #1,d1               ;Seiten-
                                       Deskriptor?
111:          beq cont                 ;ja-
112:          and #$0f,d1              ;Adresse
               ausblenden
113: cont:    or.l #$8000fc01,d1       ;ergibt
                                       Langformat
114:          move.l d1,(a1)+
115:          btst #1,d1               ;Seiten-
116:          beq next                 ;Deskriptor-
117:          move.l d2,d1
118:          clr.b d1
119:          cmp.l a2,d1              ;außerhalb
120:          bcc next0                ;der Tabelle
121:          and #$fff0,d2
122:          sub.l crpsav+4,d2
123:          add.l d2,d2              ;neue Adresse
124:          add.l d6,d2              ;berechnen
125:          bra next0
126: next:    clr.b d2
127: next0:   move.l d2,(a1)+
128:          dbra d0,copy
129:
130:          move.l a1,a0
131:          moveq #3,d0
132:          lea (384,d6.l),a2        ;Tabellen-
133: dloop:   move.l #$8000fc03,(a2)+ ;Deskriptoren
134:          move.l a0,(a2)+          ;für ST-RAM
135:          lea 256(a0),a0
136:          dbra d0,dloop
137:
138:          moveq #0,d1
139:          moveq #127,d0
140: init:    move.l #$0000fc01,(a1)+
141:          move.l d1,(a1)+
142:          add.l #$00008000,d1
143:          dbra d0,init
144:
145:          move.l $08,o_bus
146:          move.l #buserr,$08
147:
148:          pmove crpreg,crp         ;neuer
                                       Rootpointer
149:                                   ;(ATC wird
                                       gelöscht!)
150:          ptestr #7,(d5.l),#7,a0   ;Adresse der
151:                                   ;Bildschirm-
152:                                   ;Deskriptoren
153:          bset #0,2(a0)            ;Supervisor-Only
154:          bset #0,10(a0)           ;für 5 Pages
155:          bset #0,18(a0)           ;(=Videoram)
156:          bset #0,26(a0)
157:          bset #0,34(a0)
158:
159:          move.l d7,-(sp)
160:          move #SUPER,-(sp)        ;Rückkehr in
161:          trap #GEMDOS             ;User-Modus
162:          addq.l #6,sp
163:
164:          move #-1,-(sp)
165:          move.l d5,-(sp)          ;neue Adresse
166:          move.l d5,-(sp)          ;für Bildschirm
167:          move #SETSCREEN,-(sp)
168:          trap #XBIOS
169:          lea 12(sp),sp
170:          pea message(pc)          ;SCREENWATCH
171:          move #CCONWS,-(sp)       ;ist
172:          trap #GEMDOS             ;installiert
173:          addq.l #6,sp
174:          clr -(sp)
175:          pea (a6)                 ;Programmlänge
176:          move #PTERMRES,-(sp)     ;Programm
177:          trap #GEMDOS             ;resident
178:                                   ;halten
179:
180: error:
181:          pea (a5)                 ;Fehlermeldung
182:          move #CCONWS,-(sp)       ;ausgeben
183:          trap #GEMDOS
184:          addq.l #6,sp
185:          pea inserr(pc)           ;nicht
               installiert
186:          move #CCONWS,-(sp)
187:          trap #GEMDOS
188:          addq.l #6,sp
189: exit:    move.l d7,-(sp)
190:          move #SUPER,-(sp)        ;zurück in
191:          trap #GEMDOS             ;User-Modus
192:          addq.l #6,sp
193: quit:    clr -(sp)
194:          trap #GEMDOS
195:
196:
197:          dc.l "XBRA"
198:          dc.l magic
199: o_bus:   dc.l 0
200:
201: buserr:
202:          ptestr #3,([16,sp]),#7
203:          pmove psr,status         ;Adresse testen
204:          btst #5,status           ;Supervisor
205:          bne wrterr               ;Only-
206: oldbus:  move.l o_bus(pc),-(sp)
207:          rts
208:
209: wrterr:
210:          movem.l a0-a2/d0-d2,-(sp)
211:          move #7,-(sp)            ;BEL
212:          move #2,-(sp)
213:          move #BCONOUT,-(sp)
214:          trap #BIOS               ;Alarmton
               ausgeben
215:          addq.l #6,sp
216:          movem.l (sp)+,a0-a2/d0-d2
217:          bset #2,11(sp)           ;Befehl im
218:          rte                      ;Supervisor-Modus
219:                                   ;wiederholen
220:
221:
222: crpreg:  dc.l $80000003           ;alle
                                       Deskriptoren
223:          dc.l $00000000           ;im Langformat
224:
225:
226: message:dc.b $0d,$0a,"SCREENWATCH V1.0 "
227:          dc.b "installiert",$0d,$0a
228:          dc.b "1991 by Uwe Seimet",$0d,$0a,0
229:
230:
231: *Diverse Fehlermeldungen
232:
233: sterr:   dc.b $0d,$0a,"SCREENWATCH läuft nur "
234:          dc.b "auf einem TT!",0
235:
236: memerr:  dc.b $0d,$0a,"Zu wenig Speicher!",0
237:
238: inserr:  dc.b $0d,$0a,"SCREENWATCH V1.0 "
239:          dc.b "nicht installiert",$0d,$0a,0
240:
241:
242:          bss
243:
244: crpsav:  ds.l 2
245:
246: status:  ds.w 1                   ;für MMU-Status
247:
248: table:   ds.b 512+1024+15         ;Deskriptortabelle
```

ROM-Patch, der nächste

Reaktionen

Nein, es gibt heute keine neuen Patches, obwohl ich inzwischen schon wieder einige Änderungen an meinem TOS vorgenommen habe. (Boot-Sektor-Überwachung, schnelleres Schreiben auf Diskette, zusätzliche Formatiermodi, Entpacker in 'PEXEC').

Diese Änderungen können aber nicht als einfacher Patch durchgeführt werden, weil im ROM nicht genug Platz ist. Ich habe das ROM daher komplett neu assembliert.

Ich möchte heute öffentlich auf die Zuschrifften eingehen, die zu dem Artikel 'ROM-PATCH, der nächste' (Ausgabe 10/90) bei mir eingegangen sind.

Zunächts noch eine allgemeine Bemerkung zu Leserbriefen: Da die Briefe, die sich auf einen bestimmten Artikel beziehen, an den jeweiligen Autor weitergeleitet werden, ist es ganz günstig, wenn bei der Anschrift gleich 'z.Hd. <Autorenname>' angegeben werden. Die Post kann so schneller weitergeleitet werden. Sie muß dann nicht mehrmals gelesen werden. Bei Anfragen sollte Ihre Adresse nicht nur auf dem Umschlag stehen, sondern auch auf dem Brief. Bei den meisten Briefen ist dies bereits der Fall, aber es gibt immer wieder ein paar Ausnahmen.

Die meisten Zuschriften bezogen sich auf das PD-Programm von M. Rogge. Die Leser wollten entweder die gepatchte Version vor dem Brennen noch einmal testen oder eigene Änderungen vornehmen. Die aktuelle Adresse von M. Rogge habe ich nicht. Herr Rogge ist nach seinem Studium nach Großbritannien gegangen, um Auslandserfahrungen zu sammeln.

Mir ist nicht bekannt, ob das Programm von einem PD-Vertrieb angeboten wird. So lange es nicht zu viel wird, bin ich bereit, für eine angemessene Aufwandsentschädigung (10 DM inklusive Disk und Porto bei Vorkasse) eine Kopie des Programms zu verschicken. Dazu gibt es noch ein ähnliches Programm, daß ich geschrieben habe. Mein Programm reloziert das TOS direkt beim Laden. Außerdem ist es auch für die Länderversionen für F, NL und GB verwendbar.

Die Unterscheidung erfolgt nach *(_sysbase)+28*. Hier steht die Kennung für das Land. Mir sind folgende Kennungen bekannt:

```
 3 Deutschland
 5 Frankreich
 7 Großbritannien
17 Niederlande
```

Wenn sich Ihr Programm gleich in der richtigen Sprache melden soll, könnt Sie hier das Land überprüfen:

```
land%=PEEK(LPEEK(&H4F2)+28)
```

Mit dem bisher Gesagten habe ich bereits angedeutet, daß es verschiedene Versionen vom TOS 1.4 gibt. Der Unterschied zwischen den verschiedenen Versionen beruht auf verschiedenen Routinen für die nationalen Sonderzeichen der Tastatur und natürlich in den übersetzten Texten. Während die Texte alle am Ende des Betriebssystems stehen, befindet sich die Tastaturroutine zwischen $FC3E00 und $FC4000. Alle Programmteile hinter der Tastaturroutine verschieben sich daher um einen bestimmten Bereich. Die veröffentlichten Änderungen sind aus diesem Grund nur für das deutsche TOS verwendbar. Nur die Patches 5,6 und 7 können unverändert eingefügt werden. Bei den anderen Patches müssen die absoluten Adressen korrigiert werden. Da die französischen und die niederländischen Versionen länger sind als die deutsche, reicht der Platz am Ende nicht für die zusätzlichen Funktionen (Patch 4 u. 10). Man müßte das TOS komplett neu assemblieren, um Platz zu bekommen.

Ein Leser wollte wissen, ob ich Ihm die Patches nicht in sein TOS 1.2 einbauen könnte. Im Prinzip wäre dies wahrscheinlich möglich, aber Aufwand dafür ist doch recht hoch. Man müßte das TOS erst analysieren, ob die entsprechenden Routinen sich geändert haben, und wo sie überhaupt stehen. Ich habe auch kein Programm, mit dem ich aus dem (gepatchten) TOS 1.2 eine RAM-Version machen kann, damit man das Ergebnis überprüfen kann. Man sollte daher einen Zeitaufwand von einigen Tagen kalkulieren. Wenn man einen halbwegs akzeptablen Stundenlohn ansetzt, ist es erheblich günstiger, sich das neue TOS zu kaufen und - nach dem Patchen - neu zu brennen. Wer auf das alte TOS nicht ganz verzichten will (ein paar schlecht programmierte Programme laufen nicht mit dem neuen TOS), kann sich als Alternative ein umschaltbares TOS brennen. Dabei werden zwei TOS-Versionen in sechs 64kByte-EPROMs gebrannt und die höchste Adreßleitung (Pin 1) jeweils über einen Schalter auf +5V oder auf Masse legt.

Zum Schluß noch ein Punkt, der nicht nur auf meinen Beitrag zutreffen dürfte:

Es gibt immer wieder Zuschriften, bei denen jemand ein Listing abgeschrieben hat. Das Programm funktioniert aber nicht, weil sich beim Abschreiben Fehler eingeschlichen haben. Wir erhalten dann das Listing mit der Bitte, es zu überprüfen, weil der Leser die Fehler nicht finden kann.

Wie bereits in dem Kasten 'Ein Wort in eigener Sache' (Seite Leserbriefe) steht, können wir dies nicht machen, weil der Zeitaufwand dafür zu groß wird. Da die Listings direkt aus einer Kopie des Quelltextes erzeugt werden, ist es auch sehr unwahrscheinlich, daß in der Zeitung ein fehlerhaftes Listing abgedruckt wird (zumindest wenn der Autor das Programm nach einem eventuellen Zeilenumbruch noch einmal getestet hat).

Man kann also schon davon ausgehen, daß der Fehler beim Abschreiben gemacht

wurde. Für alle, die sich wenig mit Programmieren und Fehlersuche befassen, möchte ich ein paar Tips geben.

Bei der Fehlersuche kann man drei Wege gehen:

1) Man vergleicht das Listing mit der Vorlage. Diese Methode dürfte von den meisten Leuten verwendet werden. Bei dieser Methode besteht die Gefahr, daß man die Fehler übersieht, weil man nicht konzentriert genug bei der Sache ist. Dieses Vorgehen empfiehlt sich vor allem, wenn das Programm mit einer Fehlermeldung abgebrochen wird. Mit etwas Glück ist die Zeile fehlerhaft, in der der Programmabbruch erfolgte. Andernfalls sollte man sich ansehen, ob alle Variablen, die in dieser Zeile verwendet werden, überall richtig geschrieben sind.

2) Man schreibt das Programm noch einmal ab (ja ich weiß, daß ist mühsam, es kann aber die schnellste Methode sein, wenn man den Text nicht grade nach dem Adlersystem eingibt). Besser ist es meist, wenn die zweite Abschrift von einem anderen gemacht wird (nicht, weil dann der andere die Arbeit hat, sondern weil die Wahrscheinlichkeit kleiner ist, das der andere die gleichen Fehler macht). Nun vergleicht man die beiden Versionen z.B mit dem folgendem kurzen Programm. Man sollte sich nicht daran stören, daß man dabei einige Unterschiede findet, die man nicht unbedingt als Fehler interpretieren muß (z.B. eine andere Schreibweise in Infotexten). Natürlich kann man sich auch auf einen Teil des Programms beschränken, wenn man weiß, daß der Fehler in diesem Bereich liegen muß. Diese Methode wird sehr häufig in der Wirtschaft angewandt, wenn man sicher sein will, daß die eingegebenen Daten fehlerfrei sind.

```
OPEN "i",#1,"datei1"
OPEN "i",#2,"datei2"
PRINT "Fehlerhafte Zeilen:"
REPEAT
  LINE INPUT #1,a$
  LINE INPUT #2,b$
  INC i%
  IF a$<>b$
    PRINT i%,
  ENDIF
UNTIL EOF(#1)
CLOSE #1
CLOSE #2
```

3) Man kann das Programm etwas ändern, damit man den fehlerhaften Bereich stark einschränken kann. Die eigentliche Fehlersuche wird dann meist mit Methode 1 durchgeführt. Die erforderlichen Änderungen hängen dabei natürlich von dem jeweiligen Fehler ab. Meist wird man irgendwo eine zusätzliche Ausgabe für eine Variable einfügen.

Nehmen wir ein konkretes Beispiel: Bei meinem Programm hatte jemand die Fehlermeldung 'DATA nicht numerisch'. Dieser Fehler kann nur bei einem 'READ'-Befehl auftreten. Da das Programm keine besonderen Bildschirmaufbau verwendet, kann man einfach nach jedem 'READ'-Befehl die eingelesene Variable mit 'PRINT' ausgeben. Beim Auftreten des Fehlers kann man dann die entsprechende Stelle im Listing über das letzte korrekt gelesene 'DATA'-Statement suchen. In diesem Fall war ein paar Mal das Komma zwischen den 'DATA'-Statements durch einen Punkt ersetzt worden. Dies ist (bei GFA-BASIC) bei hexadezimalen Zahlen nicht erlaubt.

Ich hoffe, daß ich einigen von Ihnen mit diesen Tips etwas weiterhelfen konnte. Für alle, die die Listings nicht abtippen wollen, gibt es ja noch den Diskettenservice der ST-Computer.

Georg Scheibler

Inserentenverzeichnis



Ein Wort in eigener Sache

In den Jahren, die unsere Zeitschrift existiert, haben wir immer wieder versucht, durch die Beantwortung der bei uns eingehenden Briefe ein wenig Licht in das Dunkel zu bringen, das bei der Arbeit mit dem ATARI ST schon so manch einen aus der Fassung bringen konnte - eine Tatsache, die nicht nur Ihnen, verehrter Leser, sondern auch uns oft genug zu schaffen machte. Nichtsdestotrotz haben wir uns bemüht, die Probleme zu lösen und diverse Leserbriefe zu veröffentlichen, da wir der Meinung waren, daß die jeweilige Thematik auch einen größeren Leserkreis interessieren könnte. Trotzdem gibt es immer wieder Briefe, die wir nicht beantworten können oder dürfen. Damit Sie nicht allzusehr enttäuscht zu sein brauchen oder keine Antwort erhalten, möchten wir Sie bitten, sich an folgende Spielregeln zu halten, die sich aus unserer Erfahrung ergeben haben. Fällt Ihr Brief nicht unter die folgenden Kriterien, hat er gute Chancen, positiv beantwortet oder wenigstens als Hilferuf an unsere Leserschaft gedruckt zu werden.

1. Leider gehen immer wieder Briefe mit dem Wunsch ein, ein Produkt für diesen oder jenen Anwendungsfall vorzuschlagen, verschiedene Produkte bezüglich der Vor- und Nachteile gegeneinander abzuwägen und zu bewerten. Es ist uns aus Wettbewerbsgründen nicht erlaubt, ein bestimmtes Produkt zu favorisieren, selbst wenn wir das eine oder andere in der Redaktion überzeugt einsetzen. Wir können Sie in diesem Fall ausschließlich auf die von uns möglichst objektiven Tests und eventuell anstehende Fachmessen hinweisen. Bedenken Sie bitte, daß auch wir nicht jede Textverarbeitung, jedes Malprogramm und so weiter kennen und bestimmte Produkte dadurch in das Abseits drängen würden.

2. Oft erreichen uns Briefe, die sich positiv oder auch negativ über bestimmte Händler, Softwarehäuser oder deren Produkte auslassen. Sicherlich interessieren uns solche Bemerkungen. Bitte haben Sie aber Verständnis, daß wir weder Lob noch Tadel abdrucken dürfen, da diese Aussagen meist subjektiv sind. Anders sieht die Sache beispielsweise bei Gerichtsurteilen aus, die Sie, verehrte(r) Leser(in), erfochten haben.

3. Aufgrund der Vielzahl an Briefen, die uns täglich erreichen, sind wir leider nicht in der Lage, Programmfehler anhand von Listings oder ähnlichem zu korrigieren. Dennoch sollte ein Problem möglichst detailliert beschrieben sein, denn Ferndiagnosen sind prinzipiell sehr schwer, jedoch mit genauerer Angabe der Symptome eventuell durchführbar.

4. Von Zeit zu Zeit erreichen uns Briefe mit der Bitte, die Adresse des Lesers zwecks allgemeiner Kontaktaufnahme zu veröffentlichen. Würden wir dies in die Tat umsetzen, würde sich der Umfang des anderen redaktionellen Teils beträchtlich verkleinern. Ausnahmen stellen Leser in fernen Ländern dar, für die eine Kontaktaufnahme im eigenen Land recht schwierig ist.

Zum Schluß sollen ein paar Tips eventuell voreilig geschriebene Briefe verhindern.

1. Wenn Sie ein Problem bezüglich einer bestimmten Problematik haben oder an einem bestimmten Produkt interessiert sind, finden Sie interessante Artikel darüber eventuell in vorhergehenden Ausgaben userer Zeitschrift. Zur Auswahl eignet sich das Jahresinhaltsverzeichnis besonders gut, das immer am Jahresende in der ST Computer abgedruckt wird.

2. Sollten die Probleme mit der Handhabung eines Produktes zu tun haben, wenden Sie sich zunächst an Ihren Händler und über diesen an den Distributor beziehungsweise an das Software-Haus. Die Wahrscheinlichkeit, daß Ihnen das Software-Haus weiterhelfen kann, ist um ein Vielfaches höher als die, daß wir Ihnen helfen können.

3. Lesen Sie aufmerksam die Leserbrief-Seite. Viele Fragen wiederholen sich immer wieder, obwohl wir bestimmte Probleme schon mehrfach angesprochen haben.

14"-Monitor am ST?

Ich möchte hiermit eine technischen Frage an Sie richten: Welche Möglichkeiten bestehen, einen 14"-Monitor an einen Atari 1040 STFM bzw. an einen Mega ST 2 anzuschließen? Ich benötige dabei nur den hochauflösenden S/W-Modus (640x400 Punkte).

Axel Hosek, Wiesbaden

Red.: Einen sehr preisgünstigen 14"-Monitor bietet die Firma NEC mit ihrem „MultiSync GS"-Modell an. Dieses Gerät ist, ähnlich wie der Atari SM-124, ein monochromer Monitor, aber mit einer Bildschirmdiagonale von 14". Der große Vorteil des MultiSync GS besteht darin, daß er auch die mittlere und geringe Auflösung des ST darstellen kann, natürlich nur in Graustufen. Der Marktpreis dieses Monitors dürfte bei ca. 400,- DM liegen.

*

TT-Bild mit 60 oder 70 Hz?

Ich beabsichtige, mir in Kürze einen Atari TT zuzulegen. Leider bietet dieser Computer in der Standardausführung nur 60 Hz Bildwiederholfrequenz in der mittleren TT-Auflösung. Lediglich in Verbindung mit einem Großbildschirm (1280x 960) erzeugt der TT sein Monitorbild mit den vom ST her bekannten 70 Hz. Ist es nicht möglich (ähnlich dem 50/60-Hz-Patch beim ST), den TT auch in den geringeren Auflösungen (640x480 & 640x 400) mit 70 Hz Bildwechselfrequenz zu betreiben?

Andreas Thierfelder, Wuppertal

Red.: Eine Bildwiederholfrequenz von 70 Hz in den Auflösungen unterhalb von 1280* 960 ist schon wegen des TT-Farbmonitors nicht möglich. Dieser ist nicht in der Lage, Frequenzen größer 60 Hz zu synchronisieren. Dazu müßte Atari einen echten MultiSync-Monitor liefern. Mit einem solchen Monitor wäre es theoretisch denkbar, auch die geringeren Auflösungen mit 70 Hz Bildwechselfrequenz zu betreiben. Uns ist aber nicht bekannt, ob der TT-Videoprozessor für diesen Fall ausgelegt ist. Dokumentiert ist eine solche Möglichkeit von Atari jedenfalls nicht.

*

Langsamer durch ACCs

Vermindern speicherresidente Programme wie z.B. Neodesk, AUTO-Ordner-Programme oder Accessories die CPU-Leistung des ST? Wenn ja, wieso? Das Testprogramm Quickindex stellt bei mir immer rund 3-4% Geschwindigkeitsverlust fest, wenn solche Programme installiert sind. Bei zeitkritischen Programmen (z.B. MIDI-Anwendungen) stelle ich auch überdurchschnittlich viele Abstürze fest. Könnte auch hier die Ursache bei den speicherresidenten Programmen liegen?

Stefan Weibel, Schüpfen/Schweiz

Red.: Daß speicherresidente Programme die eigentliche CPU-Leistung herabsetzen, ist unwahrscheinlich. Die CPU läuft beim normalen ST mit 8 MHz Systemtakt, und dieser ist softwaremäßig nicht zu beeinflussen. Vielmehr rühren die Ergebnisse, die Quickindex liefert, von einer nicht ganz korrekten Testroutine her. Das Programm scheint beim Test der CPU-Geschwindigkeit nicht alle Interrupts (Unterbrechungsanforderungen, die meist in regelmäßigen Abständen von Hardware oder Programmen an die CPU gesendet wird) zu sperren. Speicherresidente Programme, die solche Interrupts in irgendeiner Weise nutzen, führen so tatsächlich zu einem Geschwindigkeitsverlust, allerdings auf Systemebene. Natürlich können dadurch andere, zeitkri-



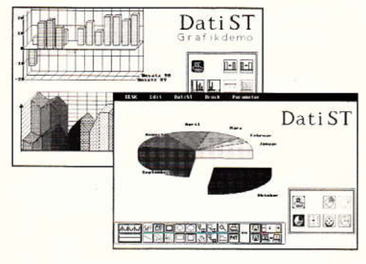


NEU

STEUERLOTSE 90

Der Steuerlotse ermöglicht die Anfertigung der kompletten, exakten Steuererklärung für jedermann (Lohn-, Gehaltsempfänger, Rentner, Gewerbetreibende usw.) für die Jahre 1984-90.
Dabei werden alle vorkommenden Daten berücksichtigt, angefangen bei Familienstand, Kindern, Kirchensteuer, Einkommen und Vermögensverhältnissen über Werbungskosten, Lebensversicherung und geleistete Vorauszahlungen bis hin zu Quellensteuer, Dividenden und Abschreibungen nach § 10e EStG.
Es gibt eben all das, was das Finanzamt sich so einfallen läßt, um an das Geld seiner braven Bürger zu kommen. Der Steuerlotse erledigt die Errechnung für Sie, gibt Tips und Erklärungen zur Materie, entscheidet, ob Lohnsteuererklärung sinnvoll, etc.
Steuerlotse wird jährlich aktualisiert; Steuerlotse 91 erscheint nach Veröffentlichung der jeweiligen Neuverordnung. Selbsterklärende Bedienung, Hilfe-Funktion.

STEUERLOTSE 90 (ST/TT[1])
SD 59 DM 30.-

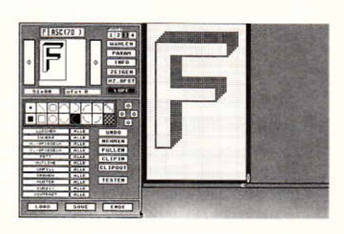


DATIST

Präsentationsgrafik

Grafiken sagen oft mehr als 1000 Zahlen, daher sollte man sich bei der Auswertung von Daten auf DatiST verlassen. DatiST stellt Ihre Daten als, Kuchen-, Reihen-, Balken-, Säulen- und Liniengrafiken dar, entweder in 2D oder 3D, gefüllt oder als Rahmen. Lage, Größe, Dehnung und der Nullpunkt einer Grafik lassen sich frei mit der Maus einstellen; dafür sorgen die iconisierten Pop-Up-Menüs. Im 3D-Modus kann gar die räumliche Perspektive frei variiert werden. Die so erzeugten Grafiken, lassen sich beschriften (z.B. mit SIGNUM!-Fonts) oder mit dem integrierten Zeichenprogramm bearbeiten, das vom Linienziehen über Blockoperationen bis hin zur Lupe alles bietet was man braucht. Um die Grafik zu Papier zu bringen bietet DatiST eine variable Druckeranpassung, die folgende Drucker unterstützt: Epson 9N/24N, NEC 24N, IBM PPR 24N, IBM AGM 24N, HP Laser, Atari-Laser!!.

DatiST (ST[1]/TT[1])
SD 40 DM 25.-



TSCHIDOS

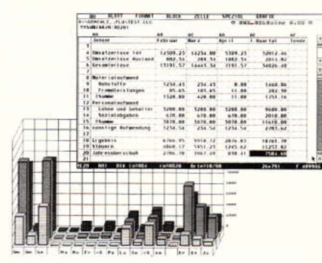
GDOS-Fonteditor

GDOS-Programme setzen sich mehr und mehr durch, doch wozu dient ein solches Programm ohne die entsprechenden Zeichensätze? Mit TSCHIDOS ist dieses Manko behoben. TSCHIDOS ist ein extrem leistungsfähiger Font-Editor mit integrierten Zeichenfunktionen (Kreisbögen, Linien, etc.) und Funktionen zum Manipulieren (Outline, Fett, Shadow, Füllen...). TSCHIDOS erzeugt Standard-GDOS-Format. Erzeugte Fonts, egal welcher Größe, können über GDOS (ASSIGN.-SYS) ins System eingebunden werden und stehen dann allen fortschrittlichen GEM-Programmen zur Verfügung. Natürlich lassen sich auch System-Fonts (z.B. für NVDI) erzeugen.

?@ABC
!"#$%&'()*
abcdefgh
äàÄöÖÜ
abc
NOPQR
0123456789

TSCHIDOS lädt neben GDOS-Fonts, auch Monostar-, STAD- und SIGNUM!-Fonts (Editor- und Drucker-Fonts), wodurch die große Welt von über 2000 Fonts erschlossen wird. Auch lassen sich Ausschnitte aus Bilddateien entnehmen.

Tschidos (ST[3]/TT[3])
SD 57 DM 30.-



GEM-CALCplus 3.0

Tabellenkalkulation

Überall dort, wo mit Zahlen hantiert wird, sei es zur betriebswirtschaftlichen Kostenrechnung, statistischen Auswertung von Meßreihen oder zur Erfassung der eigenen Finanzen, findet ein Kalkulationsprogramm seinen Einsatz. GEM-CALCplus ist ein flexibler und sehr leistungsfähiger Vertreter dieser Kategorie. Neben zahlreichen mathematischen und statistischen Funktionen bietet es eine exzellente Grafikausgabe der Daten als Kuchen-, Linien-, Balken-, Stapel-, Säulen-, Block- und Flächengrafik.

(1MB sinnvoll)

GEM-CALCplus 3.0 (ST[1]/TT[1])
SD 44 DM 25.-



Special Paint 2

Grafik de Luxe

Grafikprogramm der Extraklasse. Neben den vielen nützlichen Funktionen zeichnet sich Special Paint vor allem durch seine Geschwindigkeit, seine bequeme Bedienung und seine Kompatibilität zu bekannten Malprogrammen aus. Special Paint bietet umfangreiche Blockfunktionen, Lasso, superschnelle Lupe, Maskierungen, Clippen, schnelle Bieg-, Zerr- und Drehoptionen, Animation und vieles mehr. Clipboardunterstützung, umfangreiche Textfunktionen (ladbare Fonts, Blocksatz, Zeilenumbruch).

Special Paint (ST[1])
SD 21 DM 20.-



1stTrenn

vollautomatische Silbentrennung für 1stWordPlus

Darauf haben viele schon lange gewartet. Eine schnelle, automatische und präzise Silbentrennung für 1stWordPlus. 1stTrenn ersetzt die eingebaute Trennhilfe völlig, d.h. wird automatisch anstelle der eingebauten manuellen Trennung aktiviert (F10).

arbeitet im Hintergrund (Accessory), 1stWordPlus muß nicht verlassen werden • schnelle Trennung • wahlweise mit Bestätigung oder vollautomatisch • hohe Trefferquote von über 98% • zusätzliche Autosave-Funktion des aktiven Textes • läuft auf den deutschsprachigen 1stWordPlus Versionen 1.89, 2.02 und 3.15

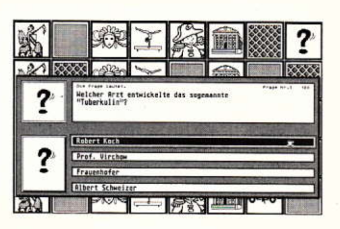
1stTrenn (ST/TT)
SD 42 DM 25.-

FORMULA

2D-/ 3D-Plotter

Für mathematisch-wissenschaftliche Anwendung. Der eingebaute Formel-Interpreter beherrscht neben allen gängigen Operationen auch die Definition verschiedener Formeln in bestimmten Teilbereichen, logische Operationen und IF..THEN..ELSE. 3D-Grafiken lassen sich aus verschiedenen Blickrichtungen anzeigen und mit Schattierungen versehen.

FORMULA (ST)
SD 23 DM 20.-



Fiffikus

Quiz mit 5000 Fragen aus 50 Wissensgebieten (Geographie, Sport, Geschichte). Ähnlich dem 'Großen Preis' werden die Fragen aus einer magischen Tafel ausgewählt, wobei FIFFIKUS vier mögliche Antworten anbietet. Untermischt mit Risiko-, Glücksfragen bietet FIFFIKUS eine abwechslungsreiche Reise in die Welt des Wissens. Von Zeit zu Zeit kann man seine Punkte auch durch ein Memory-Spiel gegen den Computer oder durch eine Superhirn-Variante aufbessern. FIFFIKUS 2 umfaßt 5000 Fragen auf 4 Disketten.
1-4 Spieler, 10-88 Jahre, s/w

FIFFIKUS 2 (ST[1]/TT[1])
SD 58 a/b/c/d DM 40.-



EASYSTAT

Induktive Statistik

EASYSTAT dient der Errechnung und Veranschaulichung statistischer Verfahren. Dabei wird neben der beschreibenden Statistik vor allem die induktive Statistik berücksichtigt. Es eignet sich für alle Anwender der Statistik (Wirtschafts- und Sozialwissenschaftler, Techniker und Studenten). Mit EASYSTAT können Daten eingegeben, dargestellt, Kennzahlen berechnet, nach Zusammenhängen gesucht und Tests bzw. Intervallschätzungen durchgeführt werden. Von zentralen Verteilungen können Quantile (oder Pseudoquantile), Verteilungsfunktion und Wahrscheinlichkeitsfunktion (bzw. Dichte) berechnet werden. EASYSTAT soll nicht zuletzt sehr abstrakte Dinge (z.B. statistische Tests) veranschaulichen helfen. Eine eingebaute einfache Kommandosprache ermöglicht es, Testprozeduren selbst zu schreiben. EASYSTAT beinhaltet einen speziell zugeschnittenen Editor und stellt ein On-Line-Hilfe-System zur Verfügung.

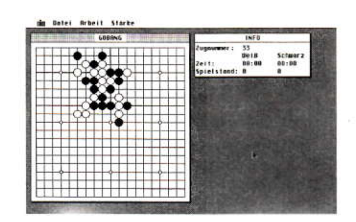
EASYSTAT (ST[1]/TT)
SD 31 DM 25.-

STatiST

modulares Statistik-Programmpaket

STatiST ist ein umfangreiches Paket zur Auswertung statistischer Daten. Zu jedem Prüfverfahren werden sämtliche Ergebnisse mit dem entsprechenden Wertungen und Kommentaren ausgegeben und, falls möglich, grafisch angezeigt. STatiST eignet sich für sämtliche, z.B. im Studium erforderlichen statistischen Auswertungen und macht das zeitaufwendige Rechnen per Hand und das Arbeiten mit Tabellen überflüssig.

STATIST (2 Disketten) (ST[1])
SD 32a/b DM 30.-

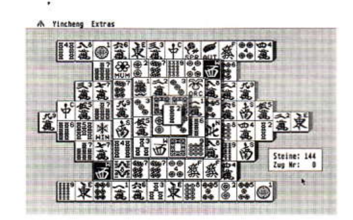


GOBANG

Ein Strategiespiel

GOBANG ist ein klassisches Brettspiel, bei dem abwechselnd Steine auf das Spielfeld gesetzt werden, wobei es gilt, 5 Steine in einer Reihe (senkrecht, waagrecht oder diagonal) zu plazieren. Der Computer bietet hier einen spielstarken Gegner, der nicht so leicht zu besiegen ist.
Neben dem Laden und Speichern einer Partie verfügt Gobang über verschiedene Spielstärken; vom Anfänger bis zum Profi. Auch die Blitzpartie, bei der jeder Spieler nur 30 Sekunden Bedenkzeit pro Spiel hat, bietet ihren speziellen Reiz. Ist man in einer schwierigen Lage, hilft der Rechner gerne mit einen Zugvorschlag aus.

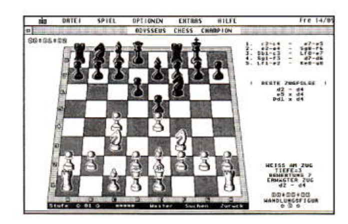
GOBANG (ST[1]/TT[1])
SD 49 DM 15.-



YINCHENG

Dieses Spiel beruht auf dem alten chinesischen Patience-Spiel Mah-Jongg. Es geht darum, das mit 144 Spielsteinen gefüllte Spielfeld zu entleeren, wobei immer nur zwei zueinander passende und nach bestimmten Regeln positionierte Steine entfernt werden dürfen. YINCHENG beinhaltet eine zwei- und eine dreidimensionale Spielvariante, die sich zwar in den Regeln, doch kaum in der Spielqualität unterscheiden.

YINCHENG (ST[1]/TT[1])
SD 45 DM 20.-



ODYSSEUS

Schachprogramm

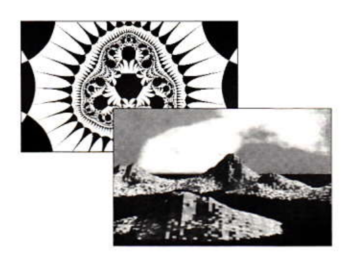
Hinter Odysseus steckt ein spielstarkes und komfortables Programm. Die Züge lassen sich leicht per Maus eingeben. Es verfügt über eine Zeit- und eine Tiefensteuerung (bis zu 12 Halbzüge) und beherrscht den Turniermodus. Die beigefügte, jederzeit erweiterbare Bibliothek erlaubt dem Programm den Zugriff auf wichtige Züge. Mit ihm kann man Partien speichern, nachspielen und analysieren lassen.

Odysseus (ST[1]/TT[1])
SD 41 DM 25.-

LITTLE SMALLTALK

Little Smalltalk ist eine Smalltalk-Implementierung, basierend auf Little Smalltalk 2 von Timothy Budd von der Oregon State University. Es eignet sich hervorragend zum Einstieg in die objektorientierte Programmierung (Vererbung von Funktionen), auf die viele Programmierer warten. Little Smalltalk hat sie. Smalltalk ist eine Sprache, die sich von herkömmlichen stark unterscheidet, so gibt es keine Datentypen, sondern nur Objekte.
Little Smalltalk-Programme sind portabel und in dieser Form auf MS-DOS- und UNIX-Systemen einzusetzen. Der Sprachschatz ist die Objektstruktur frei erweiterbar und offen. Little Smalltalk beherrscht im übrigen die Metaklassen von Smalltalk 80. Ein umfangreiches Handbuch (ASCII und TeX) beschreibt sämtliche implementierten Objekte und Primitive.

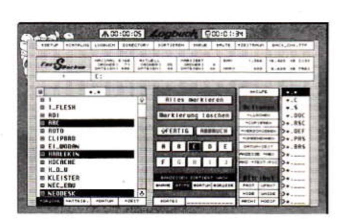
Little Smalltalk (ST)
SD 56 DM 25.-



Art Of Fractals

Expedition ins Land der Fractale. A.O.F. beginnt bei Apfelmännchen (jedoch in 3D), behandelt Julia-Mengen, Iterationen aus der Planzen und Tierwelt und entführt Sie in dreidimensionale Landschaften. Steile verschneite Gebirgshänge im Mondschein oder eine Meereslandschaft an einem wolkigen Tag? Das Programm berechnet und stellt sie dar. A.O.F. erzeugt Fantasielebewesen und läßt mathematische Pflanzen gedeihen.

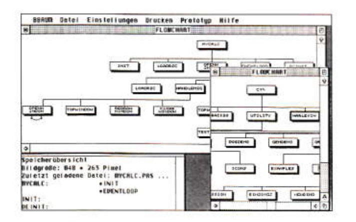
Art Of Fractals (ST[1]/TT[1])
SD 52 DM 20.-



FastSectorBackup 4.0

FastSectorBackup ist das ideale Tool für Ihre Datensicherung. Zum einen bietet es ein Image-Backup, welches komplette Partitionen sichert, und zum anderen ein sehr flexibles FileBackup. Damit lassen sich einzelne Dateien, welche nach Wildcards, Datum, Archiv-Bit oder einfach per Mausklick markiert werden, sichern. Weiterhin bietet FastSectorBackup die Möglichkeit, mehrere Backup-Vorgänge mit verschiedenen Markierungsarten in Batch-Dateien festzuhalten. Diese können dann automatisch ablaufen.

FastSectorBackup (ST[1]/TT[1])
SD 35 DM 25,-



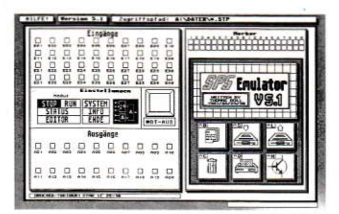
BBAUM

BBAUM ist ein äußerst leistungsstarkes Tool für die Programmdokumentation von C-, PASCAL- und GFA-BASIC-Programmen. Vor allem die Einarbeitung in fremde Quelltexte wird vereinfacht, indem grafisch in Form eines Baumes die Funktions- bzw. Prozedurabhängigkeiten dargestellt werden.

BBAUM untersucht: C-Quelltexte • PASCAL-Quelltexte • GFA-BASIC-Quelltexte (2.0, 3.0 und 3.5) • DMP-Dateien (interne Baumstruktur) • Verzeichnisse (Struktur Ihrer Festplatte/Diskette)

BBaum verwaltet Includes bzw. ausgelagerte Programmteile und fügt sie automatisch an die entsprechenden Stellen im Hauptprogramm an. Wahlweise werden auch die Routinen dargestellt, die in der System-Library definiert sind (z.B. *printf* oder *getchar*).

BBAUM (ST/TT)
SD 50 DM 25.-



SPS-Emulator V 5.1

für programmierbare Steuerungen

Unser SPS-Emulator baut auf einem SIEMENS PG 605-Programmiergerät in STEP 5 auf. Mit ihm lassen sich SPS-Programme schreiben, auf Simulationsbasis austesten, laden, speichern, ändern, ausdrucken und als FUP (Funktionsplan mit logischen Gattern) ausgeben. Enthalten sind ein Editor, ein Interpreter und FUP-Generator. Alle Befehle wurden voll im Siemens S5 Standart umgesetzt

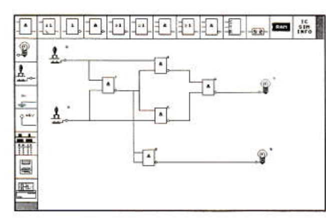
SPS Emulator V5.1 (ST[1]/TT)
SD 14 DM 25.-

Dialog Construction Set

für GFA-BASIC 3.x

Mit dem Dialog Construction Set (DCS) lassen sich auf einfache Art und Weise LST-Dateien erstellen, die den Programmcode zur Behandlung von Dialogboxen unter GFA-BASIC 3.0 enthalten. So ist es möglich, diese schnell und bequem in eigene Programme einzubauen. Als Voraussetzung wird natürlich weiterhin das Resource Construction Set (wird bei GFA-BASIC mitgeliefert) benötigt. Einfach mit dem RCS erstellen und dann mittels DCS den Programmcode generieren. Grundkenntnisse über Dialogboxen und GFA-BASIC-Programmierung sind aber weiterhin erforderlich.

DCS (ST[1]/TT[1])
SD 48 DM 15,-



ICSIM

Logik-Simulator

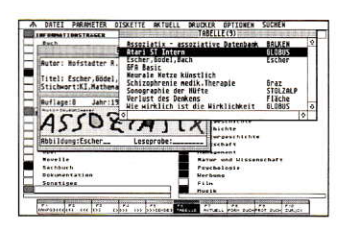
Das Programm simuliert das Verhalten von logischen Schaltungen. Bausteine und Verbindungen werden frei per Maus positioniert bzw. verbunden. Eine Schaltung läßt sich somit leicht austüfteln, testen und erst dann in die Praxis umsetzen.
Es sind die Logikbausteine nach DIN 40900 enthalten: AND, OR, NOT, NAND, NOR, XOR, RS-FF, KLEMME, LAMPE, SCHALTER, 0V und +5V.
Die Simulation wird als Impulsdiagramm oder Logiktabelle ausgegeben. Weiterhin liefert das Programm den Schaltplan und eine Liste der benötigten Bauteile.

ICSIM (ST[1]/TT[1])
SD 25 DM 20.-

DATEI LOGIK

Datenbank, die einfache Handhabung und große Flexibilität miteinander vereint. So ist es für jedermann möglich, sich ohne große Anstrengung eine Datenbank nach seinen Vorstellungen aufzubauen. Mit Hilfe des integrierten Formulareditors kann eine individuelle Abfragemaske erstellt, mit dem Etiketteneditor das Layout von Aufklebern oder Karteikarten für jeden Aufgabenbereich festgelegt und mit der Mailmerge-Funktion mit den Daten auch Serienbriefe erstellt werden.

Datei Logik (ST[1]/TT[1])
SD 36 DM 20,-



ASSOZIATIX

Assoziative Datenbank

Assoziatix ist eine assoziativ-Muster orientierte Datenverwaltung, die es ermöglicht aus einer großen Datenmenge bestimmte Gruppen auszufiltern und daraus dank schneller assoziativer Suche nach bestimmten Konstellationen, Zusammenhänge zu finden (z.B. Rasterfandung).
Mit Hilfe des Formulareditors können die Eingabemasken leicht am Bildschirm gestaltet werden, sogar mit Grafikeinbindung.

Einige Besonderheiten:
- Paßwortschutz, Export- und Importfunktion, Serienbriefe, Reportdokumentation Statistische Berechnung numerischer Werte Expertfunktion, Volltextsuche Grafikeditor: Spiegeln, Drehen, Zoomen. Balken- Linien und Kuchengrafik.

ASSOZIATIX (2 Disketten) (ST)
SD 27 a/b DM 30.-

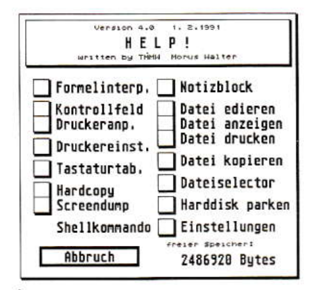


FATSPEED II

Festplattenbeschleuniger

FATSPEED II beschleunigt Festplatten unter TOS 1.0 und TOS 1.2 um bis zu 1000%, und macht damit jede Platte so schnell wie unter TOS 1.4. Bei den alten TOS-Versionen wird beim Schreiben auf Platte - hauptsächlich durch die Organisation der FAT - Zeit verschwendet, weniger durch das Schreiben an sich. Gerade bei vollen Platten (welche Platte ist schon leer?) wurden den Schreibzugriffe zur Geduldsprobe. FATSPEED II optimiert dies und erreicht somit traumhafte Schreibzeiten. Ein Restore-Vorgang für Backup-Dateien braucht z.B. keine 2 Stunden, sondern nur noch 20 Minuten, das Speichern des Desktop-Infos 2 statt 7 Sekunden, ein voller Ordner eine statt zehn Minuten. (nur sinnvoll für TOS 1.0 oder TOS 1.2)

FATSPEED II (ST)
SD 55 DM 25.-



HELP!

Multi-Accessory

HELP! besteht aus vielen nützlichen Elementen, die als Accessory in GEM-Programmen bereitstehen: Kontrollfeld, Druckeranpassung, Druckereinstellung, Datei kopieren, Editor, Notizblock, Fileselector, Harddiskpark u.a. Es verfügt über eine erweiterte Hardcopy in verschiedenen Größen und wahlweise mit Bildausschnitt (Graustufenkonvertierung bei Farbe). Ebenso kann der Bildschirminhalt auf Disk in gängigen Grafikformaten abgelegt werden. Der HELP!-Fileselector bindet sich mit Optionen zum Drucken, Formatieren, Löschen und Umbenennen ins System ein.Ein Formelinterpreter, auch Taschenrechner genannt, ermöglicht die Berechnung komplexer Formeln in binär, octal, hex und dezimal, verfügt über Variablen, logische Verknüpfungen und viele mathematische und trigonometrische Funktionen. Ein kleiner GEM-Editor in eigenem Fenster hilft beim schnellen Anzeigen oder Ändern von Texten. Darüber hinaus können alle Kommandos einer externen Shell von HELP! aus aufgerufen werden.

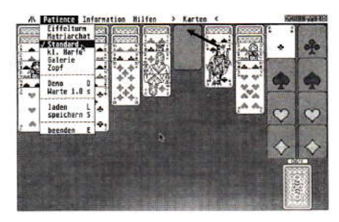
HELP!
SD 54 DM 25.-

➪ Sonderdisk-Bestellung

Sonderdisks können Sie telefonisch oder schriftlich bestellen, oder nutzen Sie einfach die Bestellkarte im Heft.

Bei Nachnahme zzgl. DM 4.- Gebühr, Versandkosten DM 5.- (Ausland DM 10.-)

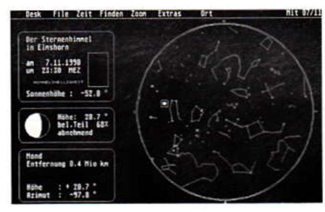
MAXON Computer
Schwalbacher Str. 52
W-6236 Eschborn
Tel: 06196/481811



PATIENCE

Das Patiencespiel (patience = franz.: Geduld) stammt aus Frankreich. Es ist ein Kartengeduldsspiel, das hohe Aufmerksamkeit erfordert. Es schult das Denkvermögen, fördert die Kombinationsfähigkeit, entspannt und beruhigt zugleich. Im Programm sind folgende Patience-Varianten enthalten: Standard, Eiffelturm, Zopf, Kleine Harfe, Matriarchat und Bildergalerie. Patiencen verfolgen das Ziel, Karten nach bestimmten Regeln sortiert abzulegen. Sind alle Karten abgelegt, gilt die Patience als gelöst. Das Programm gibt auf Wunsch Lösungsvorschläge. Eine ausführliche Anleitung zu den Patiencen fehlt ebenfalls nicht.

Patience (ST[1]/TT[1])
SD 11 DM 15,-



ST-HIMMEL

Mit dem Programm kann der Anblick des Sternenhimmels für verschiedene Orte und Zeitpunkte berechnet werden. Ein ideales Programm für den Hobby-Astronomen. Es zeigt alle mit bloßem Auge (bei gutem Wetter) sichtbaren Sterne (~3000) mit Bezeichnungen, Helligkeiten und Entfernungen • die mit bloßem Auge sichtb. Planeten • den Mond mit seiner Phase • die hellsten Sternhaufen und Nebel • einen Kometen • die Höhe der Sonne über oder unter dem Horizont • die Namen der sichtbaren Planeten • die verschiedenen Sternbilder • den Tierkreis • die Eigennamen von 190 Sternen (z.B. Großer Bär statt Ursa Maior) • die Tag- und die Nachtseite der Erde auf einer Weltkarte.

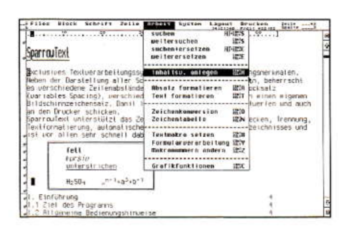
ST-HIMMEL (ST[1])
SD 38 DM 20,-

DAME

Computerumsetzung des alten Brettspiels, wobei der ST einen spielstarken Gegner darstellt. Die Figuren werden per Maus angewählt, die Züge protokolliert und analysiert. Verschiedene Spielstärken, Zugvorschläge, Laden und Speichern einer Partie, sowie verschiedene Spielvarianten dürfen nicht fehlen.

DAME (ST[4]/TT)
SD 29 DM 15.-
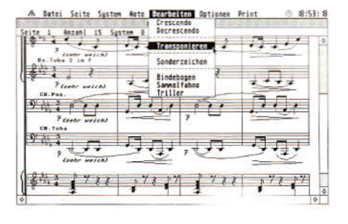
[1] nur für Monochrommonitor (SM124=640*400 bzw. 'ST Hoch')
[2] nur für Farbmonitor
[3] alle monochrome Auflösungen
[4] nicht für Mega STE (TOS 2.x)



SparrowText

Exklusives Textverarbeitungssystem mit besonderen Leistungsmerkmalen. Neben der Darstellung aller Schriftarten auf dem Bildschirm beherrscht es verschiedene Zeilenabstände, Proportionalschrift im Blocksatz (variables Spacing), verschiedene Font-Größen und vor allem einen eigenen Bildschirmzeichensatz. Damit lassen sich Sonderzeichen entwerfen und auch an den Drucker schicken.
SparrowText unterstützt das Zeichnen von Linien und Rechtecken, Trennung, Textformatierung, automatische Erzeugung eines Inhaltsverzeichnisses und ist vor allem sehr schnell dabei.
Als besonderen Leckerbissen ermöglicht es Formularverarbeitung, die sich hervorragend zum Ausfüllen von Briefbögen, Adreßfeldern oder allgemeinen Formularen eignet. Die Eingabefelder lassen nach Wunsch auch Eingabebeschränkungen (z.B. nur Zahlen) zu und bieten daher die Möglichkeit, gewisse Felder miteinander aufzuaddieren. Weiterhin kann man diese Felder automatisch ausfüllen lassen, da SparrowText Daten von einer Datenbank importieren kann und diese in die Felder einträgt. Dadurch läßt sich das Programm für Serienbriefe, Zeugnisse oder gar Rechnungen/Mahnungen einsetzen.

SparrowText (ST[1]/TT[1])
SD 37 DM 25,-

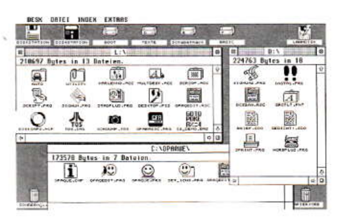


TRISTAN

Notensatzsystem

Für alle Musikfreunde, die nicht nur vom Blatt spielen, sondern auch aufs Blatt schreiben, bietet das Notensatzsystem TRISTAN die ideale Möglichkeit, ihre Noten professionell zu Papier zu bringen. Es lassen sich Partituren mit bis zu 100 Seiten mit max. 32 Notensystemen je Seite bearbeiten. Alle im klassischen Notensatz gebräuchlichen Zeichen lassen sich bequem mit der Maus edieren. Ebenfalls stehen mehrere Notenschlüssel, Sammelfahnen, Triller und Bindebögen zur Verfügung. Automatische Transponierfunktion. Ausdruck auf 9- und 24-Nadeldruckern, im 24-Nadelmodus in *maximaler* Druckerauflösung.

TRISTAN (ST/TT)
SD 24 DM 25.-



OPAQUE

Das Desktop mit neuem Gesicht

Wie wäre es mit einem zweckmäßigen und originellen Desktop? Opaque bietet die Möglichkeit, jedem Programm ein eigenes,sinnbezogenes Icon zuzuordnen. Auch die Laufwerke lassen sich ändern. Weiterhin kann man die Icons mit Wildcards definieren. Samt Icon-Editor und über 100 Icons.

OPAQUE (ST)
SD 22 DM 15.-

Programmierer aufgepaßt!!

Haben Sie nicht auch ein Programm geschrieben, das in diese Serie paßt? Sonderdisketten enthalten leistungsstarke Programme aus allen Bereichen zu günstigen Preisen. Als Autor erhalten Sie eine attraktive Umsatzbeteiligung. Lassen Sie doch mal was von sich hören.

MAXON Computer
Idee Sonderdisk
Industriestr. 26
W-6236 Eschborn

Weitere Sonderdisks

Nr.	Titel	System	DM
01	TOS	nicht mehr lieferbar	
02	RCS	ST/TT	15,-
03	Extended VT52	ST[1]	15,-
04	Lovely Helper	ST/TT	15,-
05	Accessories	ST	15,-
06	NIKI	ST[1]	15,-
07	VirusEx	ST/TT[3]	15,-
08	Ariadne	ST[3]/TT[3]	15,-
09	Legende	ST[2]	15,-
10	Quinemac	ST[1]/TT[1]	15,-
12	MagicBox ST	ST/TT	15,-
13	Robotwar	ST[1]/TT	15,-
15	Hardcopy II	ST/TT	15,-
16	Easy Adress	ST[1]/TT[1]	15,-
17	IconDesign	ST[1]/TT[1]	15,-
18	Panda	ST[1]	15,-
19	MAKI	ST[1]/TT[1]	15,-
26	Hauskasse	ST[1]	15,-
28	Master Etikett	ST[1]/TT[1]	15,-
30	Würfelpoker	ST	15,-
33	Ultra-Disk	ST	15,-
34	Fußball	ST[1]/TT[1]	15,-
39	Länder der Welt	ST[1]/TT[1]	15,-
43	Koala	ST[2]	15,-
46	Take_1	ST[1]/TT[1]	15,-
47	Complex	ST[1]/TT[1]	20,-
51	OrdneHBD	ST	20,-
53	Chipcopy	ST[1]/TT	25,-



SONDERDISK

Sonderdisks beinhalten Programme aus den verschiedensten Bereichen (z.B. Utilities, Grafik, Schulung, Spiele). Sonderdisks ermöglichen den Usern, qualitativ hochwertige Software zu einem kostengünstigen Preis zu erhalten. Im Preis ist eine Beteiligung der Autoren enthalten.

In der nächsten ST-Computer lesen Sie unter anderem

Calamus SL

"Was lange währt, wird endlich gut!" Ob dieser Satz auch für das lang angekündigte Calamus SL gilt, werden wir Ihnen in der nächsten Ausgabe berichten können. Ein Programm oder vielleicht besser Programmsystem wie Calamus SL läßt zumindest einiges an Leistungsfähigkeit erwarten.

Handy Scanner

Passend zum DTP- oder Bildbearbeitungsprogramm werden für den Atari diverse Handyscanner angeboten. In den letzten Jahren haben sich diese Geräte doch ganz schön gemausert. 32 Graustufen etc. sind dabei keine Seltenheit, so daß sie sich durchaus für alltägliche Scan-Vorgänge in ansprechender Qualität eignen. Wir stellen Ihnen eine Auswahl vor.

Grafikkarten

Grafikkarten haben derzeit auf den Atari-Rechnern Hochsaison. Wir wollen in unserem Test zwei Karten vorstellen, die aufgrund Ihres Preis-/Leistungsverhältnisses auch für den "normalen" ST-Besitzer recht interessant sind. Es handelt sich dabei um die Imagine-Karte der Firma Wittich und die Crazy Dots-Karte von TKR.

Lernprogramme

Früher wurde man von Eltern, Geschwistern usw. beim Lernen unterstützt. Heute macht das unser elektronischer Freund mit einer unglaublichen Beharrlichkeit, bei der obige bereits einen Nervenzusammenbruch gehabt hätten. Grund genug für uns, sich mal auf dem Software-Markt für Lernprogramme umzuschauen.

Die nächste ST-Computer erscheint aufgrund unserer Sommerpause erst am Fr., dem 30.08.91

Fragen an die Redaktion

Ein Magazin wie die ST-Computer zu erstellen, kostet sehr viel Zeit und Mühe. Da wir weiterhin vorhaben, die Qualität zu steigern, haben wir Redakteure eine große Bitte an Sie, liebe Leserinnen und Leser: Bitte haben Sie Verständnis dafür, daß Fragen an die Redaktion nur **donnerstags von 14⁰⁰-17⁰⁰ Uhr** unter der Rufnummer 06196/481814 telefonisch beantwortet werden können.

Natürlich können wir Ihnen **keine** speziellen Einkaufstips geben. Wenden Sie sich in diesem Fall bitte an einen Fachhändler. Wir können nur Fragen zur ST-Computer beantworten.

Vielen Dank für Ihr Verständnis!

Impressum ST Computer

Chefredakteur: Harald Egel (HE)

Redaktion:
Harald Egel (HE)
Dieter Kühner (DK)
Joachim Merz (JM)
Christian Möller (CM)

Redaktionelle Mitarbeiter:
C.Borgmeier (CBO)
Claus Brod (CB)
Ingo Brümmer (IB)
Derek dela Fuente (ddf)
Thorsten Luhm (thl)
U.Seimet (US)
R.Tolksdorf (RT)
Thomas Werner (TW)

Autoren dieser Ausgabe:
R.Blittkowsky
J.Bolt
D.Brockhaus
M.Chakravarty
R.Darr
C.Dernbach
G.Ekart
U.Hax
A.Hollmann
Dr.M.Kester
C.Kluss
B.Krönung
U.Mast
R.Peiler
F.van Mengen
B.Rosenlecher
G.Scheibler
M.Srowig
J.Starzynski

Auslandskorrespondenz:
D.Dela Fuente (UK)

Redaktion: MAXON Computer GmbH
Postfach 59 69
Industriestr. 26
6236 Eschborn
Tel.: 0 61 96/48 18 14, FAX : 0 61 96/4 11 37

Verlag: Heim Fachverlag
Heidelberger Landstr. 194
6100 Darmstadt 13
Tel.: 0 61 51/5 60 57, FAX : 0 61 51/59 10 47 + 5 60 59

Verlagsleitung:
H.J.Heim

Anzeigenverkaufsleitung:
U.Heim

Anzeigenverkauf:
K.Sterna, H. Arbogast

Anzeigenpreise:
nach Preisliste Nr.6, gültig ab 2.1.91
ISSN 0932-0385

Grafische Gestaltung:
Manfred Zimmermann

Titelgestaltung:
Axel Weigend

Fotografie:
Andreas Krämer

Illustration:
Manfred Zimmermann

Produktion:
B.Kissner

Druck:
Frotscher Druck GmbH

Lektorat:
V.Pfeiffer

Bezugsmöglichkeiten:
ATARI-Fachhandel, Zeitschriftenhandel, Kauf- und Warenhäuser oder direkt beim Verlag

ST Computer erscheint 11 x im Jahr
Einzelpreis: DM 8,-, ÖS 64,-, SFr 8,-
Jahresabonnement: DM 80,-
Europ. Ausland : DM 100,- Luftpost : DM 130,-
In den Preisen sind die gesetzliche MWSt. und die Zustellgebühren enthalten.

Manuskripteinsendungen:
Programmlistings, Bauanleitungen und Manuskripte werden von der Redaktion gerne angenommen. Sie müssen frei von Rechten Dritter sein. Mit seiner Einsendung gibt der Verfasser die Zustimmung zum Abdruck und der Vervielfältigung auf Datenträgern der MAXON Computer GmbH. Honorare nach Vereinbarung. Für unverlangt eingesandte Manuskripte wird keine Haftung übernommen.

K-SPREAD 4

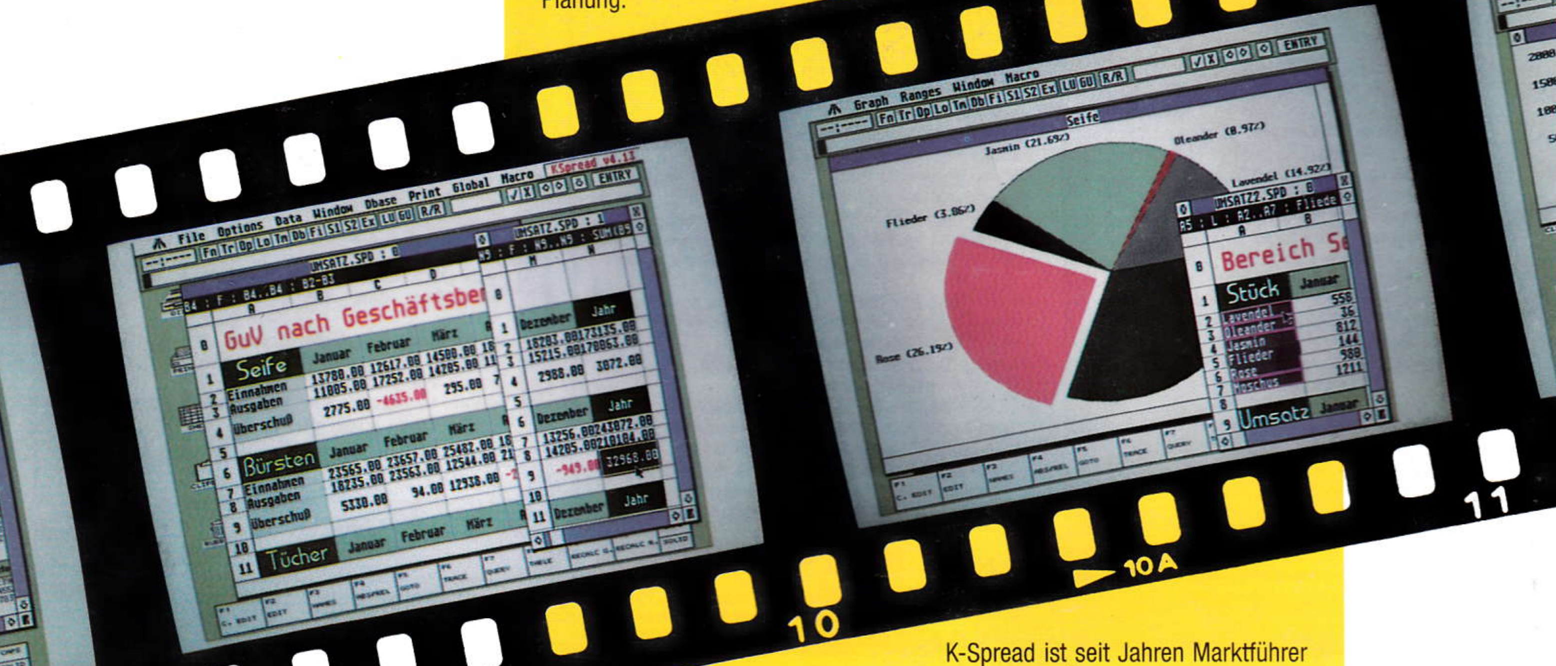
TABELLENKALKULATION SCHAFFT DURCHBLICK

Eine Tabellenkalkulation ist ein elektronisches Rechenblatt. Sie geben alle Ausgangsdaten und die Zusammenhänge ein, z.B. Umsatz = Stückzahl × Preis. Wenn Sie jetzt das Feld mit dem Preis ändern, berechnet K-Spread 4 automatisch den dadurch veränderten Umsatz.

Da auch umfangreiche Rechenblätter so in Sekundenschnelle durchgerechnet werden, schaffen »was..wäre.. wenn«-Überlegungen mit K-Spread 4 schnell fundierte Grundlagen für Ihre Planung.

Die Benutzeroberfläche ist konsequent GEM-Standard. Bei vielen Funktionen kommen Sie so von alleine darauf, wie sie funktionieren.

Als einzige Tabellenkalkulation auf dem ST arbeitet K-Spread 4 mit bis zu acht Fenstern. Durch »herüberziehen« können Sie somit blitzschnell Daten zwischen verschiedenen Rechenblättern übertragen.



K-Spread 4 läuft auf allen Atari ST und TT (ab 1 MB). Es unterstützt sogar Farbe und Großbildschirme. Wie Sie sehen, wird die Farbe auch genutzt. Präsentations-Grafiken werden in Farbe viel klarer; und negative Zahlen kann K-Spread automatisch rot darstellen.

K-Spread ist seit Jahren Marktführer in England. Über 10.000 Stück wurden dort verkauft. Die deutsche Version wurde von OMIKRON für deutsche Erfordernisse maßgeschneidert. K-Spread 4 erhalten Sie bei allen OMIKRON.Vertragshändlern oder direkt bei OMIKRON. Unverbindliche Preisempfehlung

DM 248,–.

OMIKRON.Soft- + Hardware GmbH
Sponheimstr. 12a · D-7530 Pforzheim
Telefon 07231/356033



XEST, Webgasse 21, A-1060 Wien
OMIKRON, France, 11, rue dérodé, F-51100 Reims
Elecomp, 11, avenue de la gare, L-4131 Esch/Alzette
Jotka Computing, Postbus 8183, NL-6710 AD Ede